

Enhanced OS-9 for X86 PCAT

Version 1.2



MICROWARE™

Intelligent Products For A Smarter World

Copyright and Publication Information

Copyright ©2000 Microware Systems Corporation. All Rights Reserved. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from Microware Systems Corporation.

This manual reflects version 1.2 of Enhanced OS-9 for X86.

Revision: B
Publication date: February 2000

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, Microware will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction Notice

The software described in this document is intended to be used on a single computer system. Microware expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of Microware and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

For additional copies of this software/documentation, or if you have questions concerning the above notice, please contact your OS-9 supplier.

Trademarks

OS-9, OS-9000, DAVID, and MAUI are registered trademarks of Microware Systems Corporation. SoftStax, FasTrak, UpLink, MICROWARE TECH-CHECK, and Hawk are trademarks of Microware Systems Corporation. All other product names referenced herein are either trademarks or registered trademarks of their respective owners.

Address

Microware Systems Corporation
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Table of Contents

Chapter 1: Installing and Configuring Enhanced OS-9

7

8	Introduction
8	Host System Requirements
8	Target System Requirements
10	Support and Documentation
10	Applicable Documents
10	OS-9 Product Support
11	Microware TECH-CHECK
12	Installing OS-9 on the Windows Host Computer
12	Hardware Setup
12	Memory
12	Disk Drives
12	Monitor
13	Keyboard/Mouse
13	Ethernet Support
14	CMOS Settings
15	Installing OS-9 on the Target X86 Hardware
15	Configuring the Hardware and Basic CMOS Settings
16	Making a Boot Floppy
21	Preparing the Hard Disk
23	Advanced Configurations
33	Finishing the Hard Disk Configuration
33	MAUI Graphics Support
38	Cross-Hosted Development Using Hawk
38	Starting Hawk on the Target
38	Running MAUI Graphics
39	Running the Hawk Tutorial
39	Project File Creation

40	Customizing the Component
44	Building the Project
44	Loading and Executing the HomeWatch Application
45	Security
46	Modifying the Key Code
47	Using Cross Hosted Utilities
47	os9dir
47	os9dcheck
48	os9list
49	Additional Cross Hosted Utilities
51	Supported Devices
52	Ethernet Controllers
53	3COM PCI
57	3COM ISA
60	SMC Ultra 83C790
64	SMC 8390
64	System State Debugging - Not Supported
67	3COM PCMCIA
70	DEC 21140
73	AM79C961 & AM79C73A
74	NE2000
78	Cirrus Logic CS8900
81	Sequential Device Support
81	VGA Graphics / Keyboard
82	Serial Mouse
83	PS2 Mouse
83	16550 Serial
84	Digiboard
85	HostessI
86	Risicom
87	Parallel Printer
88	Physical Disk Media
88	IDE Standard

93	PCMCIA IDE
99	IDE Descriptors
101	DiskOnChip
106	DiskOnChip Descriptors
107	PC AT Style Floppy
108	Floppy Descriptors
108	Symbios 810,810A,825,825A and 875 PCI SCSI controllers—Wide, Ultra and Ultra Wide
114	Diamond FirePort20 and FirePort40—Wide, Ultra and Ultra Wide
119	Adaptec 1540/1542 ISA
119	Adaptec 2940, 2940U and 2940UW
119	SCSI Descriptors
120	System Devices
120	Real Time Clock
121	Additional Devices
121	PPP and SLIP
123	X86 Utilities
123	ABORT
123	CACHECHK
123	DMPPCI
124	GIMMEIO
125	LOOP
126	MOUSE
126	PCIV
128	PCMCIA
130	PINFO
130	SETPCI
131	SYMBIOS_INFO
136	TESTPCI
137	VIDBIOS
138	ROM Utilities and Special Booters
138	llkermi

138	llcis
139	rpciv
140	PCI Configuration Information
140	PCI Library User Guide
140	_pci_search_device() - search for PCI device
141	_pci_next_device() - find next PCI device
142	pci_get_config_data() - get PCI configuration data
144	pci_find_device() - find PCI device
145	pci_find_class_code() - find PCI device based on class code
146	pci_read_configuration_byte() - read PCI configuration byte
147	pci_read_configuration_word() - read PCI configuration word
148	pci_read_configuration_dword() - read PCI configuration dword
149	pci_write_configuration_byte() - write PCI configuration byte
150	pci_write_configuration_word() - write PCI configuration word
151	pci_write_configuration_dword() - write PCI configuration dword
152	pci_get_irq_pin() - get PCI IRQ pin
152	pci_get_irq_line() - get PCI IRQ line
153	pci_set_irq_line() - set PCI IRQ line
155	Hawk Profiler

Chapter 1: Installing and Configuring Enhanced OS-9

This chapter describes installing and configuring Enhanced OS-9 for X86. It includes the following sections:

- **Introduction**
- **Support and Documentation**
- **Installing OS-9 on the Windows Host Computer**
- **Installing OS-9 on the Target X86 Hardware**
- **Cross-Hosted Development Using Hawk**
- **Using Cross Hosted Utilities**
- **Supported Devices**
- **X86 Utilities**
- **PCI Configuration Information**
- **Hawk Profiler**

Introduction

This manual provides information to help you get started with Enhanced OS-9 for the X86. OS-9 is an architecturally advanced, high performance real-time operating system available for the IBM/Motorola, Intel, SuperH, MIPS and ARM/StrongARM microprocessor families. Refer to the documentation on the Enhanced OS-9 for the X86 CD for additional OS-9 product information.

Enhanced OS-9 for X86 includes the OS-9 stand-alone microkernel for X86; Hawk, Microware's cross-hosted development solution; a copy of Microware's new configuration builder tool, graphics, networking and complete on-line documentation.

Host System Requirements

- Microsoft Windows 95, 98 or NT
- Pentium Processor
- 32 MB RAM
- 250 – 350 MB Free Disk Space
- CD-ROM Drive
- Network Card (required when using Microware's Hawk to debug applications on the target computer)

Target System Requirements

- PCAT compatible computer
- Monitor (may be removed once OS-9 is installed)
- IDE, SCSI or other storage device
- Floppy drive (may be removed once OS-9 is installed)

- Network connection to the host computer (required for initial configuration and when using Microware's Hawk to debug applications)
- Keyboard or serial connection to the Windows host computer (may be removed once OS-9 is installed)
- Optional mouse

Support and Documentation

Applicable Documents

The OS-9 Product Documentation Set for X86 is included on the CD-ROM and may be optionally installed on the Windows host computer. The following OS-9 manuals are referenced by this getting started manual:

OS-9 Utilities Reference Manual

OS-9 Product Support

Microware stands behind its products with an experienced staff of Technical Support engineers, available to assist you with your development projects from 8:00 AM to 6:00 PM CST. Whether you have installation, configuration, basic usage, or advanced questions, our team is poised to take on the challenge.

Telephone support is available by calling 515-224-0458. You may also send your questions to support@microware.com or post your questions on our bulletin board at bbs.microware.com.

Product release notes are available at <http://www.microware.com/x86relnotes>.

Microware TECH-CHECK

Microware TECH-CHECK is a state-of-the-art call tracking system that maintains a complete record of your product concerns and questions. You can assist us in handling your questions more effectively by using Microware TECH-CHECK when contacting Customer Support.

Microware TECH-CHECK is a program wizard which asks you a series of questions about your system, your questions/concerns, and your contact information. Microware TECH-CHECK creates a text file, which can be mailed to support@microware.com.

You can access Microware TECH-CHECK by selecting *Microware TECH-CHECK* from the Start->Programs->Enhanced OS-9 for X86 folder.

Installing OS-9 on the Windows Host Computer

Microware's Enhanced OS-9 for the X86 must be installed on your Windows 95/98 or NT computer, prior to running OS-9 on your target machine. Insert the OS-9 CD-ROM into your CD-ROM drive and select the installation option from the autorun menu. Follow the on-line instructions to complete the install.

The installation creates an "Enhanced OS-9 for X86" program folder and includes full documentation, Microware's Hawk development environment, Microware's Configuration Wizard and the OS-9 for X86 binary images.

The package may be removed by selecting "Uninstall OS-9 for X86" from the program menu.

Hardware Setup

Memory

OS-9 will run with as little as 2 MB of RAM, however, it may be more convenient to install additional memory when developing and testing graphic intensive applications.

Disk Drives

While OS-9 can be configured as a ROM or flash based system, these instructions assume that a floppy drive and IDE hard disk will be used for initial development and testing.

Monitor

A monitor may be connected to the target system for graphics applications and to aid in the initial OS-9 configuration.

Keyboard/Mouse

Connect a keyboard and PS/2 or serial mouse to the target system. Both the keyboard and mouse are optional, although a keyboard should be used when initially installing OS-9.

Ethernet Support

Development work will be facilitated with the addition of an ISA, PCI or PCMCIA network card. The network card must be supported by OS-9. Additional cards are constantly being added, so check bbs.microware.com for the latest list. Driver support is included for network cards from the following manufacturers:

- 3COM
- SMC
- DEC/Intel
- AMD
- And various NE2000 support

Refer to the **Ethernet Controllers** section, of this document, for information on specific network cards.

Commonly available cards, supported by OS-9, include:

- Ready LINK COMPEX RL2000-PCI (use ne2000 Configuration Wizard setting)
- Etherlink XL PCI 3C900B-TPO
- Asanté Fast 10/100 PCI Adapter for the Mac & PC (use Dec 21140 Configuration Wizard setting)

For some Ethernet cards, the I/O base address and interrupt settings must be configured on the card to match the settings used by OS-9. A setup disk, provided with the network card, may be needed to configure the card to the correct settings. The default settings for an NE2000 card are I/O Base 0x340 and IRQ 9. Refer to the **Ethernet Support** section of this document for configuration settings for other cards.

CMOS Settings

It may be necessary to modify the BIOS settings in CMOS to boot from a hard disk. In most cases you may modify the CMOS settings by pressing DEL after rebooting. Configure the board with the correct settings for the attached peripherals. The boot sequence should try floppy first, and then the IDE hard drive.

Installing OS-9 on the Target X86 Hardware

Installation of OS-9 on the target hardware follows these basic steps:

1. Configuring the hardware and basic CMOS settings.
2. Making a basic boot floppy using Microware's wizard.
3. Booting OS-9.
4. Modifying the OS-9 boot image to take advantage of the peripherals on the target machine.
5. Loading the hard disk.
6. Making the hard disk bootable.
7. Configuring graphics support.

These instructions assume that:

- You have installed Enhanced OS-9 for the X86 on your Windows 95/98 or NT host computer.
- You have an X86 target computer configured with a monitor, keyboard, floppy drive and hard drive.
- You have a network connection between the target computer and the Windows host computer.
- You have a serial connection between the target computer and the Windows host computer, and have a terminal emulation program, such as Hyperterm, running on your Windows computer. Note, the serial connection is optional, but may be necessary if you would like to move the OS-9 console to a serial port so that graphics applications may have dedicated use of the display/keyboard.

Configuring the Hardware and Basic CMOS Settings

Refer to the previous section for information on configuring the hardware and CMOS settings.

Making a Boot Floppy

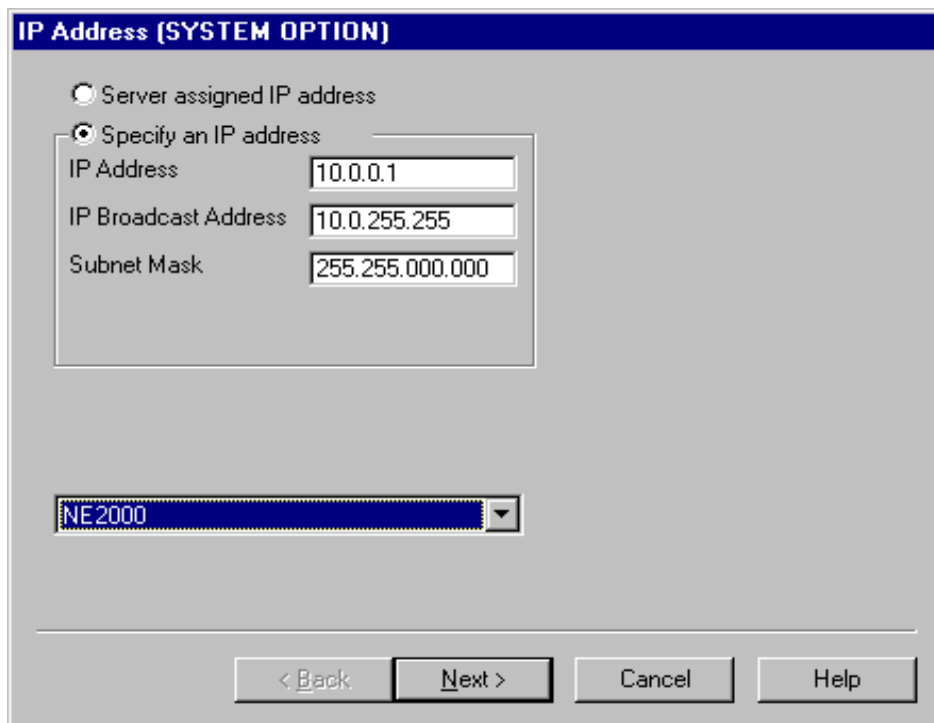
The quickest way to bring OS-9 up on the target system is to attach a floppy drive and use Microware's Configuration Wizard to build a bootable floppy disk.

- Step 1. Once OS-9 is installed on your Windows host computer, select *Microware Configuration Wizard* from the Programs->Enhanced OS-9 for X86 folder. The following dialog box appears:



- Step 2. Verify that:
- The MWOS location reflects the location of the MWOS directory tree installed on your Windows host computer.
 - The *Use Wizard* radio button is selected.
 - The *Port Selection* drop down box displays PCAT

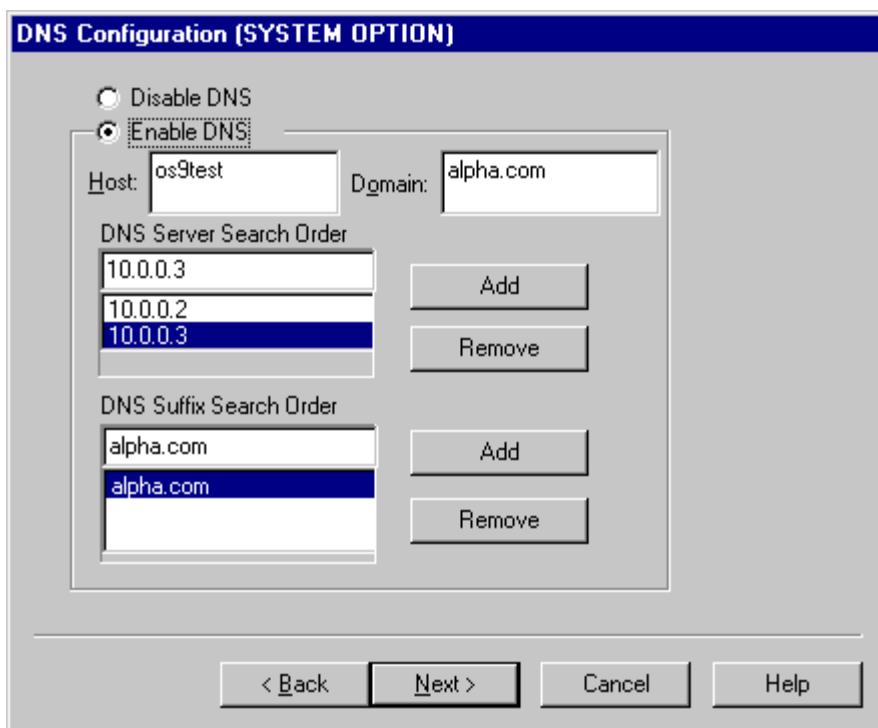
- Step 3. Type a name in the *Configuration Name* drop down box (OS9test in this example) and press OK. The IP Address dialog box appears.



The image shows a dialog box titled "IP Address (SYSTEM OPTION)". It has two radio buttons: "Server assigned IP address" (unselected) and "Specify an IP address" (selected). Below the radio buttons are three text input fields: "IP Address" with the value "10.0.0.1", "IP Broadcast Address" with the value "10.0.255.255", and "Subnet Mask" with the value "255.255.000.000". Below these fields is a drop-down list box showing "NE 2000". At the bottom of the dialog are four buttons: "< Back", "Next >", "Cancel", and "Help".

- Step 4. The IP Address Dialog box configures networking on the target. Select your network adaptor model from the drop down list box to enable networking, otherwise select *none*.
- Step 5. Select either the *Server assigned IP address* or the *Specify an IP address* radio button. Enter the IP address, broadcast address and subnet mask, if *Specify an IP address* was selected. Click *Next* to continue. The DNS Configuration dialog box appears.

- Step 6. Select the *Enable DNS* radio button if DNS is to be used. Fill in the appropriate values for your network. Select *Next* to continue.



DNS Configuration [SYSTEM OPTION]

☐ Disable DNS
☒ **Enable DNS**

Host: Domain:

DNS Server Search Order

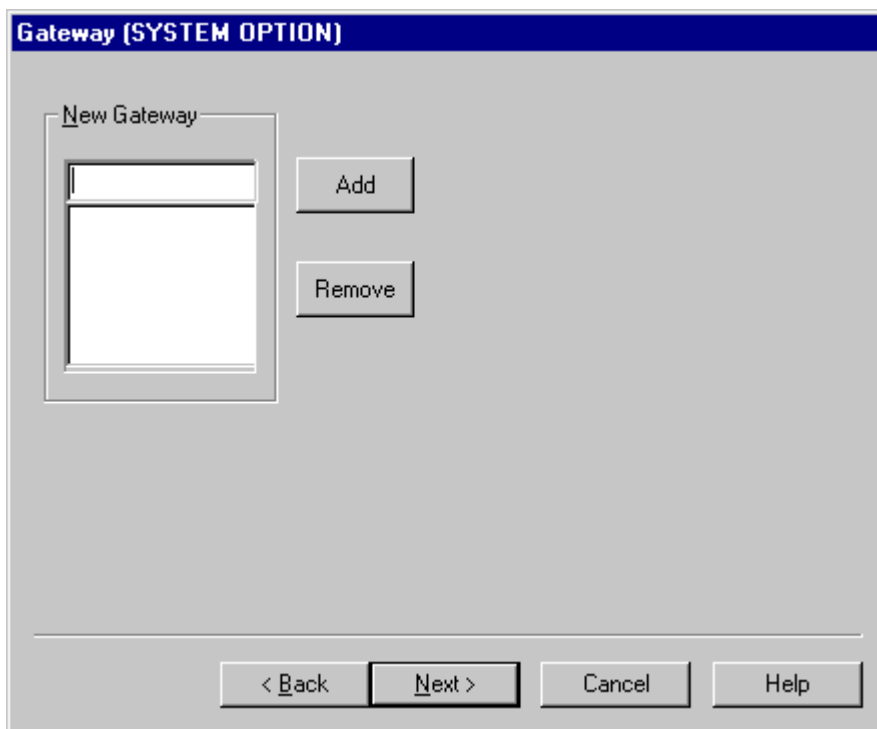
10.0.0.3	Add Remove
10.0.0.2	
10.0.0.3	

DNS Suffix Search Order

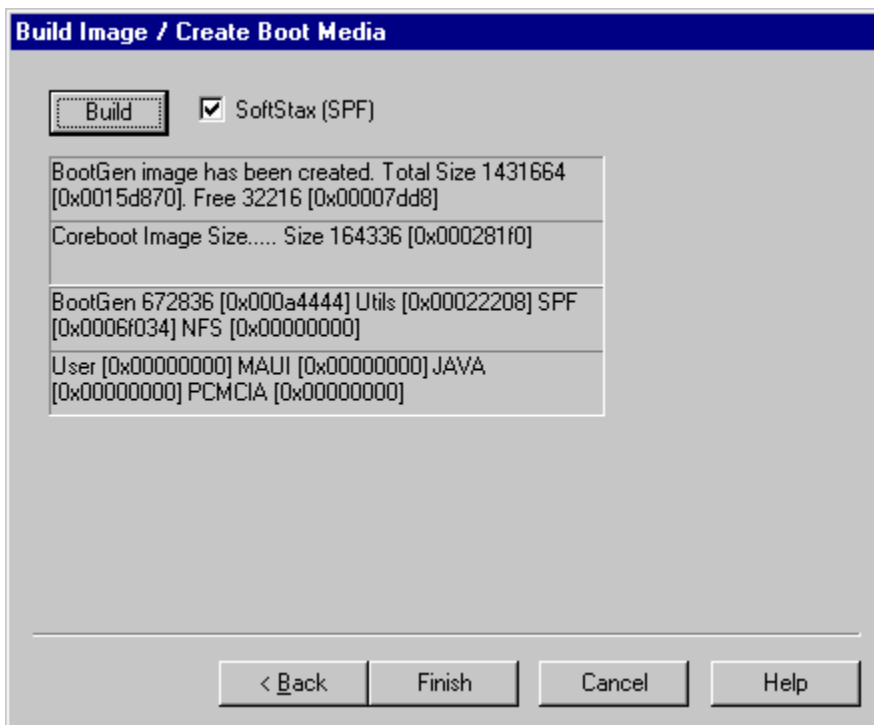
alpha.com	Add Remove
alpha.com	

< Back Next > Cancel Help

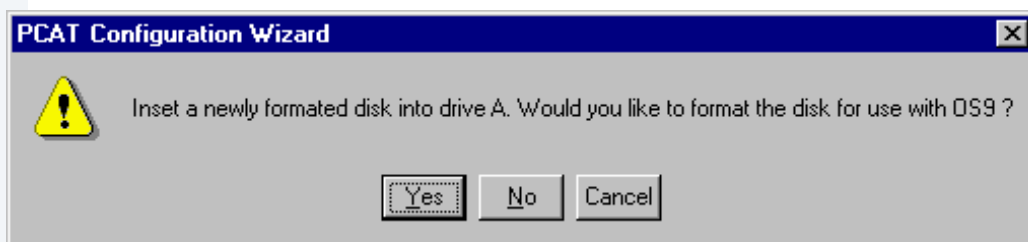
- Step 7. Enter the gateway, if appropriate for your network and click *Next* to continue.



- Step 8. Enable the *SoftStax (SPF)* radio button if using networking. Select the *Build* button to create the OS-9 boot image.



Insert a new floppy disk into your computer's A drive and click *Yes* to format it for OS-9 and copy the boot image. Select *Finish* and save your changes, when prompted.



The newly created boot floppy can be used to bring OS-9 up on the X86 target hardware. The default OS-9 console is the target computer's monitor. If networking was enabled, it should be possible to *telnet* into

the target, from the windows host computer. For example, select *run* from the Windows start menu and type *telnet OS9test*. Login as *super* with the password *user*.

Preparing the Hard Disk

The newly created boot floppy may be used to format a local hard disk with the OS-9 file system. A network connection between the OS-9 target machine and the Windows host computer may be used to load the OS-9 system files onto the hard disk. This section assumes that:

- You have built a boot floppy, with networking enabled, as described in the previous section.
- You have added a hard disk to your target machine and have used the BIOS setup to configure the disk as the primary or master drive.
- You have a network connection between the Windows host computer and the OS-9 target system.

These instructions step you through copying OS-9 system files to your hard disk. The section, *Finishing the Hard Disk Configuration*, discusses the final steps for making the disk bootable.



Note

The following instructions assume a minimal configuration with limited disk space (less than 4MB). If additional disk space is available, then you may wish to download *mw86.tar* in place of *mw86sm.tar*. *mw86.tar* includes the full command set and descriptors.

-
- Step 1. Boot the target system, using the floppy made in the previous section.
- Step 2. Run *fdisk* from the OS-9 console.
- ```
$ fdisk -d=/hcfmt -e
```
- Create OS-9000 type partition.
  - Make sure the partition is set to active.

- Make sure to use MBR option if this is a new disk with no other OS on it.

Step 3. Run format from the OS-9 console to create the OS-9 RBF file system

```
$ format /hclfmt
```



## Note

Note that physical format and physical verifies are typically not necessary.

Step 4. Download the required system files.  
On the Windows host computer, open a command window and change into the RESIDENT directory on the CD-ROM. Start an ftp session with the target machine.

```
ftp <target>
ftp> User: Super
ftp> Password: User
ftp> bin
ftp> cd /hclfmt
ftp> send tar
ftp> send diskcache
ftp> send mw86sm.tar
ftp> quit
```

Step 5. Turn on disk cache support, from the OS-9 console.

```
$ chd /hclfmt ; load -d diskcache ; diskcache -e /hclfmt=1024k
```

Step 6. Expand the system files.

```
$ load -d tar ; tmode nopause ; tar xvpf mw86sm.tar
```

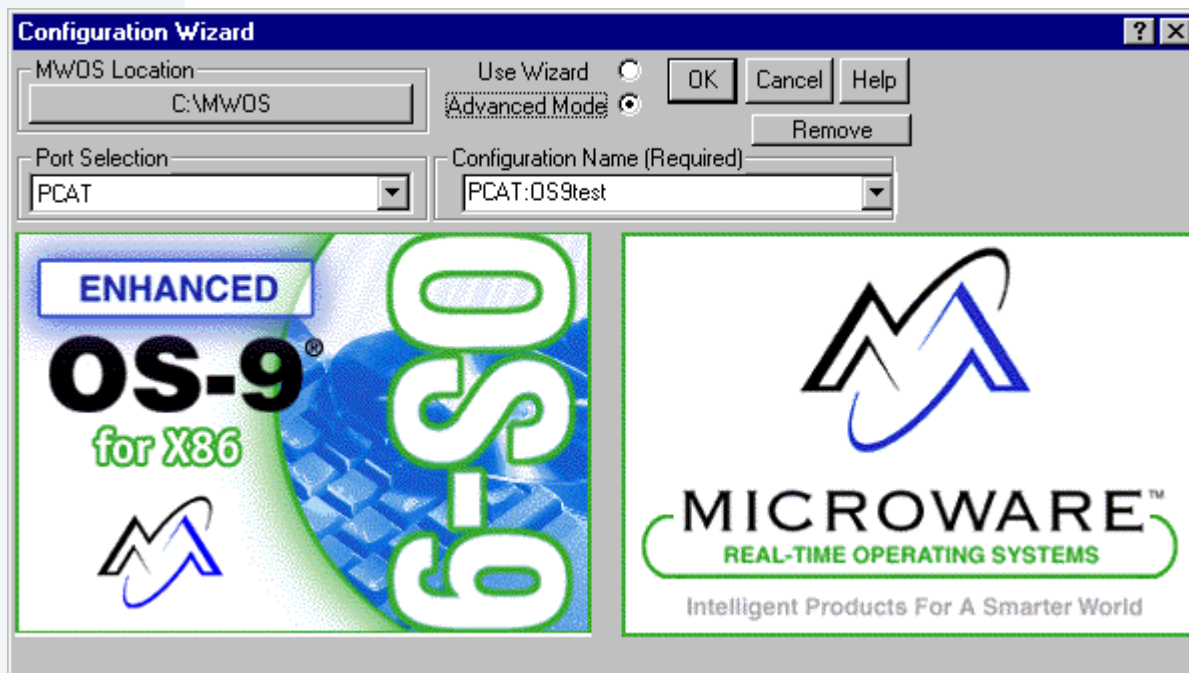
At this point, the disk has been formatted and the OS-9 system files have been copied to disk. The next section, *Advanced Configurations*, discusses building and installing an OS-9 boot file on the hard disk.

## Advanced Configurations

It may be desirable to configure the target computer to boot from a local hard disk and to move the OS-9 console to a serial port so that the monitor may be removed or dedicated to graphics applications. This section assumes that:

- You have built a boot floppy, with networking enabled, as described in the previous section.
- You have a hard disk attached to the OS-9 target system and have followed the instructions in the previous section to format and load it.
- You have the COM1 port connected to your windows host computer's serial port, with the appropriate cable, and have a terminal emulation program, such as Hyperterm running. Note, this step is optional, but may be necessary if you would like to move the OS-9 console to a serial port so that graphics applications may have dedicated use of the display/keyboard.

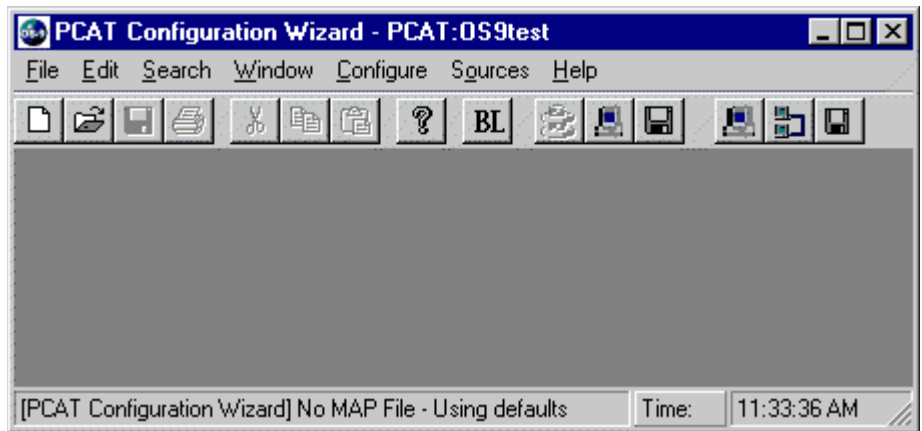
- Step 1. Select *Microware Configuration Wizard* from the Programs->Enhanced OS-9 for X86 folder. The following dialog box appears:



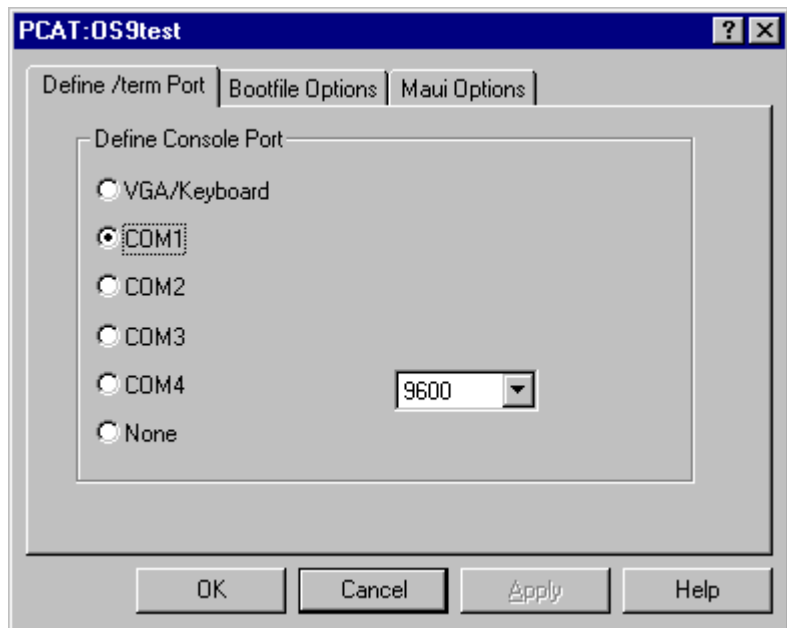
- Step 2. Click the *Advanced Mode* radio button and select the configuration used when making the boot floppy (OS-9test in this example) from the *Configuration Name* drop down list. Click *OK* to continue.



Step 3. Select Configure->Bootfile->Configure System Options from the menu.

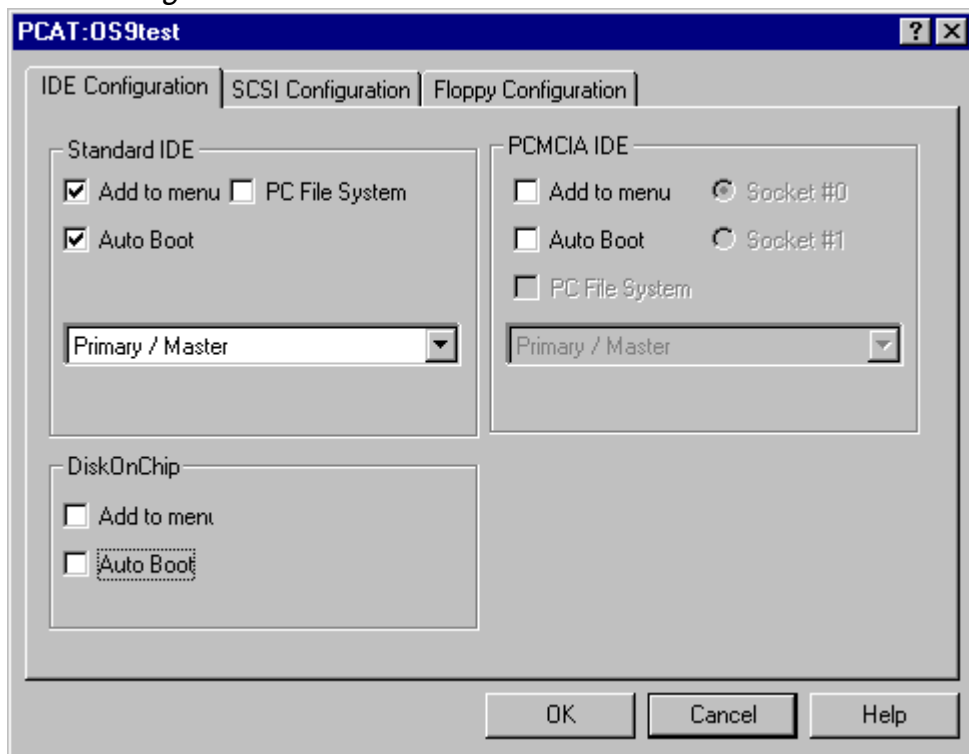


Step 4. Click on the *Define /term Port* tab.



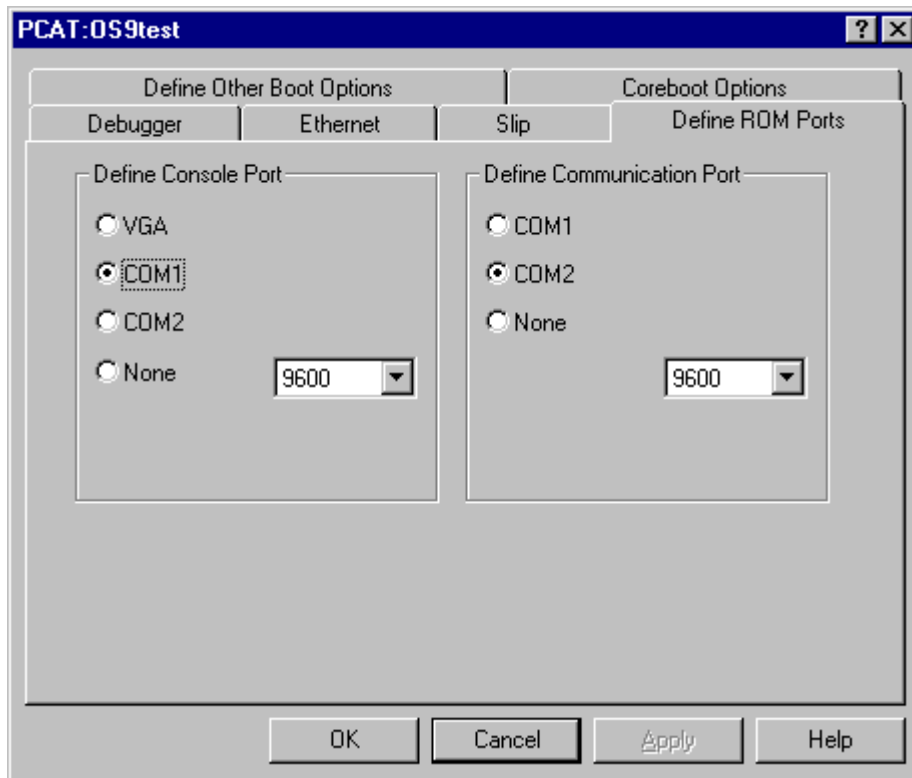
Step 5. Click on the *COM1* radio button to define the high-level console as using serial port 1. Verify that the baud rate is set to 9600 baud. Click *OK* to continue. Note, leave the high-level console at VGA/Keyboard if the monitor and keyboard will be used as the OS-9 system console.

- Step 6. Select Configure->Coreboot->Disk Configuration from the menu. Select the *IDE Configuration* tab.



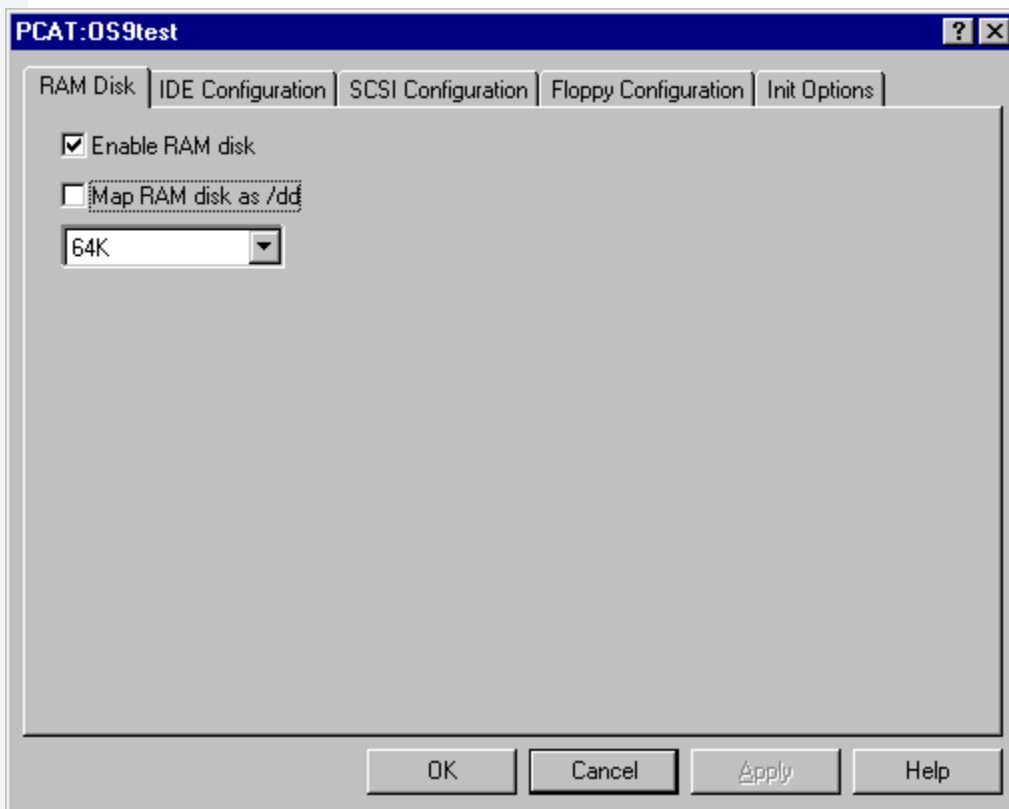
- Step 7. Enable the Auto Boot checkbox and click *OK* to continue.

- Step 8. Select Configure->Coreboot->Main Configuration from the menu. Select the *Define ROM Ports* tab



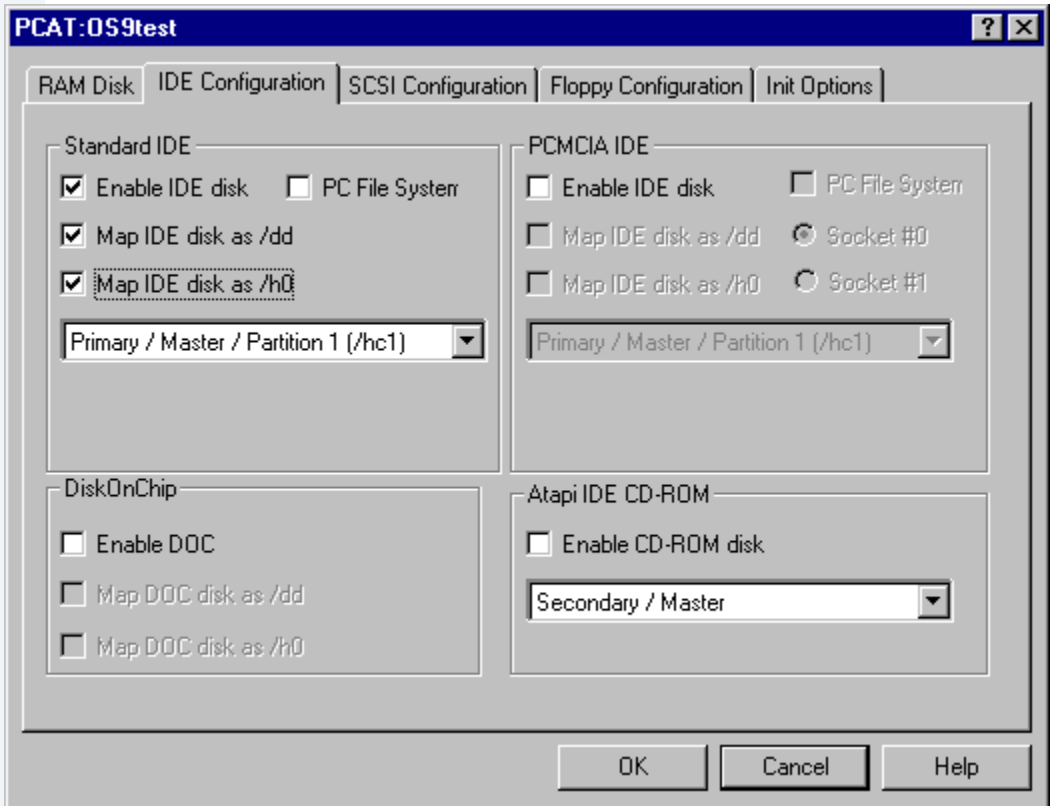
- Step 9. Select the Define Console Port *COM1* radio button and verify that the baud rate is set to 9600. Click *OK* to continue. This dialog box moves the low-level OS-9 console to serial port 1. Note, leave the low-level console at VGA if the monitor and keyboard will be used as the OS-9 system console.

Step 10. Select Configure->Bootfile->Disk Configuration from the menu. Click the *RAM Disk* tab.



Step 11. Verify that **only** the *Enable RAM disk* checkbox is checked. Select the RAM disk size from the drop down list box. Use of a RAM disk is optional, and you may disable it by clearing the *Enable RAM disk* checkbox. If enabled, the RAM disk may be accessed as /r0 on the target system.

Step 12. Select the *IDE Configuration* tab.



Step 13. Click on the *Enable IDE disk*, *Map IDE disk as /dd* and *Map IDE disk as /h0* checkboxes to enable them.

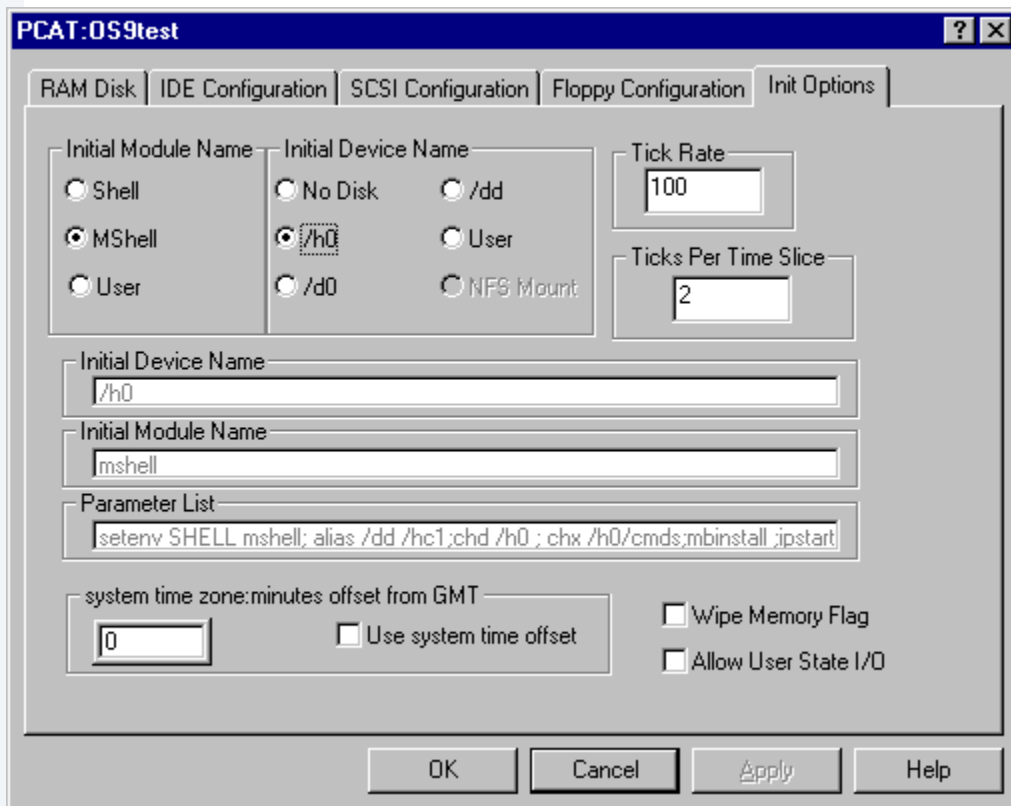


## Note

The standard IDE hard disk will be accessed as device `/hc1` from the OS-9 console. The same device may also be accessed as `/h0` or `/dd`.

An IDE CD-ROM drive may be attached to the target system and accessed as device `/cd0`. The CD-ROM must be the master device on the second IDE channel.

Step 14. Click on the Init Options tab.



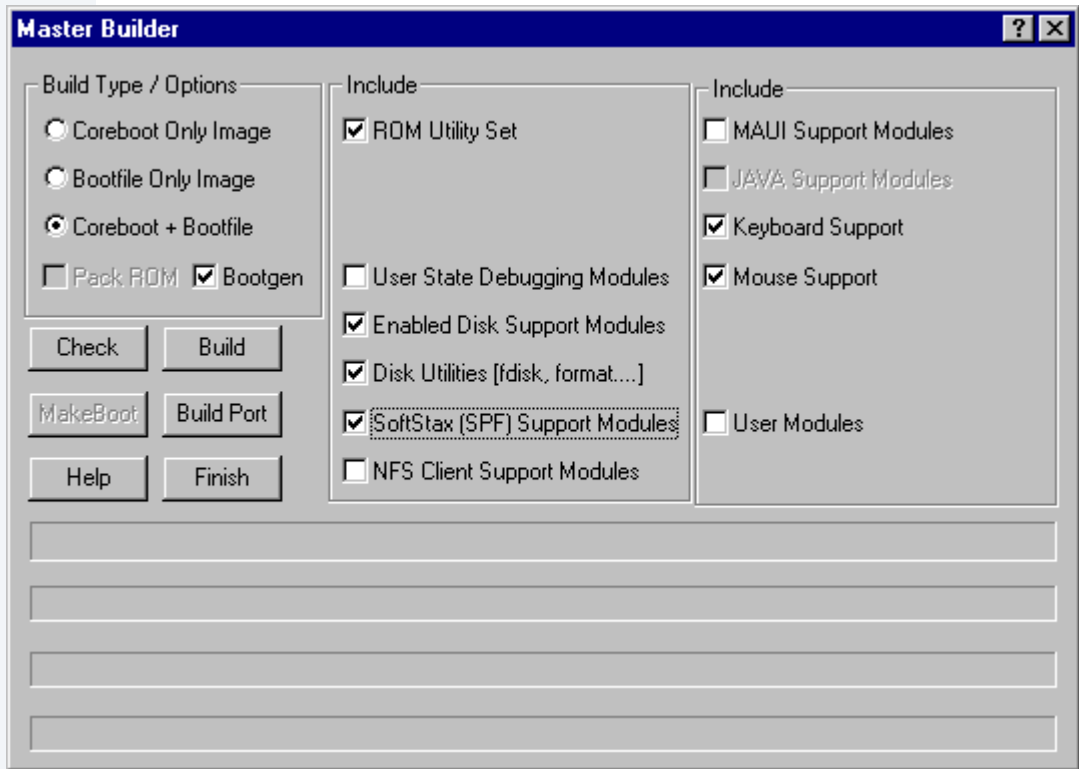
The image shows the 'PCAT:OS9test' dialog box with the 'Init Options' tab selected. The dialog has several sections for configuration:

- Initial Module Name:** Radio buttons for Shell, MShell (selected), and User.
- Initial Device Name:** Radio buttons for No Disk, /h0 (selected), /d0, /dd, User, and NFS Mount.
- Tick Rate:** A text box containing '100'.
- Ticks Per Time Slice:** A text box containing '2'.
- Initial Device Name (text box):** Contains '/h0'.
- Initial Module Name (text box):** Contains 'mshell'.
- Parameter List (text box):** Contains 'setenv SHELL mshell; alias /dd /hc1;chd /h0 ; chx /h0/cmds;mbinstall ;ipstart'.
- system time zone:minutes offset from GMT:** A text box containing '0'.
- Use system time offset:** An unchecked checkbox.
- Wipe Memory Flag:** An unchecked checkbox.
- Allow User State I/O:** An unchecked checkbox.

At the bottom are buttons for OK, Cancel, Apply, and Help.

Step 15. Select the `/h0` radio button to use the as the initial device. Click *OK* to continue.

Step 16. Select Configure->Build Image from the menu.



Step 17. Verify that the following options are enabled:

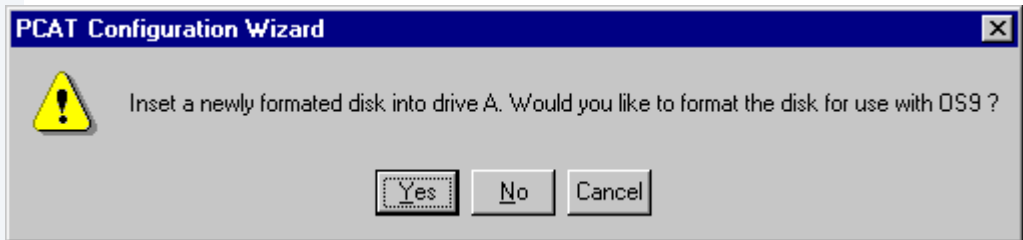
- Coreboot + Bootfile radio button
- Bootgen
- ROM Utility Set
- Enabled Disk Support Modules
- Disk Utilities
- SoftStax (SPF) Support Modules
- Mouse Support (Enables support for a PS/2 style mouse)
- Keyboard Support

**Note**

Select the *User State Debugging Modules* checkbox to include the Hawk debugging modules on the target system. Alternately, you may load and run the modules from the hard disk on the target.

Step 18. Click *Build* to create the OS-9 boot image.

Step 19. Click *Makeboot* when the image is built. The following dialog box appears:



Step 20. Insert a new floppy disk into your computer's A drive and click *Yes* to format it for OS-9 and copy the boot image. Select *Finish*, once the boot image has been written to floppy. Exit the Configuration Wizard by selecting *Exit* from the file menu. Save your changes when prompted.

Step 21. Test the new boot image by inserting the floppy into the drive on the target system and rebooting. Verify that:

- 1.OS-9 boots, with the console appearing in the correct location, either the monitor or the serial port.
- 2.You can telnet into the OS-9 computer, from the Windows computer. For example, select *run* from the Windows start menu and type *telnet OS9test*. Login as *super* with the password *user*.



## Finishing the Hard Disk Configuration

This section finishes the hard disk configuration by using the OS-9 bootgen utility to install a boot image onto the hard disk. It is assumed that the instructions in the section *Preparing the Hard Disk* were followed to format and load the disk. The hard disk will be made bootable using the OS-9 boot image created in the previous section, [Advanced Configurations](#).

- 
- Step 1. Boot the target system, using the floppy made in the previous section.
  - Step 2. Verify that you can access the hard disk. For example, from the OS-9 console execute the command:  

```
$ dir /h0
```

An OS-9 directory listing should be displayed.
  - Step 3. Turn disk caching off prior to running bootgen. At the OS-9 console, type the command:  

```
$ diskcache -d /h0lfmt
```
  - Step 4. Bootgen the new system. At the OS-9 console, type:  

```
$ bootgen /h0lfmt -i=/d0/iplhdnoq -l=/d0/firstboot /d0/sysboot -nb400
```
  - Step 5. Remove the floppy from the drive and reboot the system. The system should boot from the hard disk, with the OS-9 system prompt appearing on the console.

## MAUI Graphics Support

MAUI is Microware's graphics solution. To start MAUI from the OS-9 console, change into the /h0/sys directory and type: loadmaui. Verify that MAUI is running by executing a couple of the demo programs, such as fdraw or fcopy from the OS-9 console.

The following code fragments, from the loadmaui file, configure OS-9 for Generic VGA mode 13 graphics support. Video mode 13 works with most every graphics card, but does not provide the best resolution. You

may want to comment out the mode 13 driver by placing an asterisk in front of each line, and uncomment one of the other video drivers such as the generic VESA driver or the ISA Bank driver.

On the OS-9 target system you may edit the loadmaui file using the umacs editor. Refer to the *Using the uMACS utility* chapter in the *Utilities Reference* manual. The *Utilities Reference* manual is included with the X86 Product Documentation and may be installed on the Windows system or read directly from CD-ROM.

```
*
* Graphics card selections.
*
* Note: The cdb default is PS2 mouse. To use serial mouse
* select the "_s" version.
*
* MAUI port - Generic VGA mode 13 (320x200x8bpp)
Remove the leading asterisk from one cdb_ file, vga and gx_vga
files to enable Generic VGA mode 13 video.
*
load -d CMDS/BOOTOBJS/MAUI/cdb_vga - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_s - Serial mouse
load -d CMDS/BOOTOBJS/MAUI/vga
load -d CMDS/BOOTOBJS/MAUI/gx_vga
*
* MAUI port - Generic VGA mode 12 & "X" (640x480x4bpp &
360x480x8bpp)
Remove the leading asterisk from one cdb_ file and the vga_ext
and gx_vga_ext files to enable Generic VGA mode 12 video.
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_ext - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_vga_ext_s - Serial mouse
*load -d CMDS/BOOTOBJS/MAUI/vga_ext
*load -d CMDS/BOOTOBJS/MAUI/gx_vga_ext
*
* MAUI port - CL-GD5434 (up to 1024x768x24bpp)
Remove the leading asterisk from one cdb_ file and the gfx and
gx_cl543 files to enable graphics support for the Cirrus Logic
5434

*load -d CMDS/BOOTOBJS/MAUI/cdb - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_s - Serial mouse
*load -d CMDS/BOOTOBJS/MAUI/gfx
*load -d CMDS/BOOTOBJS/MAUI/gx_cl543
```

```

*
*

* VESA driver - uses INT 10h calls
Remove the leading asterisk from the following files to enable
VESA driver support

* the CDB determines which drivers are used. Pick one
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_vesa - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_vesa_s - Serial mouse
*
* The graphics descriptor.
*load -d CMDS/BOOTOBJS/MAUI/vesa - Must uncomment this line to
use the VESA driver
*
* The graphics driver. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/gx_vesa - Normal VESA driver
*load -d CMDS/BOOTOBJS/MAUI/gx_vesal - Linear mode VESA
*load -d CMDS/BOOTOBJS/MAUI/gx_vesah - 15 bit color support
*load -d CMDS/BOOTOBJS/MAUI/gx_vesalh - Linear mode, 15 bit
color

* ISAbank driver. Banked mode driver uses a data module
* the CDB determines which drivers are used. Pick one
*
*load -d CMDS/BOOTOBJS/MAUI/cdb_svga - PS/2 mouse
*load -d CMDS/BOOTOBJS/MAUI/cdb_svga_s - Serial mouse
*
* The graphics descriptor.
*load -d CMDS/BOOTOBJS/MAUI/svgab - uncomment to use the ISA
bank driver
*
* The graphics driver. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank1 - 1024x768 default
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank6 - 640x480 default
*load -d CMDS/BOOTOBJS/MAUI/gx_isabank8 - 800x600 default
*
* The data module. Pick one.
*load -d CMDS/BOOTOBJS/MAUI/ibcl5422 - Cirrus Logic 5422 ISA
LB
*load -d CMDS/BOOTOBJS/MAUI/ibcl5428 - Cirrus Logic 5428 VESA
LB
*load -d CMDS/BOOTOBJS/MAUI/ibcl5429 - Cirrus Logic 5429 VESA
LB

```

```
*load -d CMDS/BOOTOBJS/MAUI/ibct65548ts110cs - Toshbia 110CS
laptop
*load -d CMDS/BOOTOBJS/MAUI/ibct65550ts205cds - Toshbia 205cds
laptop
*load -d CMDS/BOOTOBJS/MAUI/ibct65550ts205cdsvga - Toshbia
205cds laptop with VGA monitor
*load -d CMDS/BOOTOBJS/MAUI/ibtlet4000 - Tseng Labs ET4000 ISA

* End of bootlist
```

The gx\_vesa driver comes in four different modules.

gx\_vesa is the normal one. It has 640x480, 800x600, 1024x768, and 1280x1024 support at 256 colors. The VESA BIOS is asked what modes are supported. The BIOS should stop the driver from setting any modes that can't be displayed. This driver will use a linear display buffer if the VESA BIOS is version 2.0 or greater and tells the driver that linear buffers are supported.

gx\_vesal will only work on cards with BIOS's that support linear mode. It is about 3 times faster than gx\_vesa on supported hardware.

gx\_vesah adds support for 15 bit high color modes. It looks for the highest resolution high color mode. The color depth table is separate from the resolution table.

gx\_vesalh is a linear buffer only with high color support.

The descriptor is vesa and there are two cdb's. cdb\_vesa and cdb\_vesa\_s. cdb\_vesa is set up for a bus mouse and cdb\_vesa\_s is set up for a serial mouse.

The gx\_isabank driver comes in three modules depending on what default resolution you want. gx\_isabank1 has a default of 1024x768, gx\_isabank6 has a default of 640x480 and gx\_isabank8 has a default of 800x600 all at 256 colors. The gx\_isabank driver uses a data module to tell it how to talk to different hardware. The data modules include are:

```
ibcl5422 cirrus logic 5422 ISA card ibcl5428 cirrus logic 5428 VESA LB
card ibcl5429 cirrus logic 5429 VESA LB card ibct65548ts110cs
Toshiba laptop 110cs ibct65550ts205cds Toshiba laptop 205cds
ibct65550ts205cdsvga Toshiba laptop 205cds with external VGA
monitor ibtlet4000 Tseng labs ET4000 ISA card
```

The descriptor is `svgab` and the cdb's are `cdb_svgab=` and `cdb_svga_s`.  
`cdb_svga` is for a bus mouse and `cdb_svga_s` is for a serial mouse.

All modules are in

`MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBSJS/MAUI`

## Cross-Hosted Development Using Hawk

---

Hawk is a comprehensive software development and management environment designed to increase the efficiency of development under OS-9. Hawk's highly integrated tool set simplifies and automates the tasks of creating, debugging, analyzing and managing complex real-time software development projects.

Hawk is much more than just an “edit-compile-debug” package. Its expanded functionality addresses all phases of product development – from initial code creation to software version control. This focus on the entire product lifecycle means reduced time-to-market and easier project management.

Microware Hawk is included with the OS-9 Board Level Solutions and Embedded Systems packages. A fully functional, sixty day evaluation version is included with the OS-9 evaluation CDs.

This example assumes that a disk based OS-9 system was prepared, as discussed in the previous sections.

### Starting Hawk on the Target

Hawk communication and debugging modules must be started on the target, prior to using Hawk. The user state debugging modules may be started from the OS-9 console by changing into the `/h0/sys` directory on the hard disk and executing `startndpd_spf`.

```
$ chd /h0/sys
$ startndpd_spf
```

### Running MAUI Graphics

The Hawk tutorial requires MAUI graphics. From the OS-9 console, change into the `/h0/sys` directory and type `loadmaui`, if you have not already done so.

## Running the Hawk Tutorial

Included on the CD is a sample MAUI application intended to introduce both MAUI programming and the Hawk IDE to the new OS-9 user. The application, named *security*, is actually one program in a series of possible applications grouped together to provide a home security suite called HomeWatch.

The purpose of this document is to show you how to (a) create a Hawk project file for the MAUI application called HomeWatch, (b) compile the security application, and (c) run the application on the target.

### Project File Creation

In order to create a project file for the HomeWatch system, it is necessary to know where the source code for the application is located. This tutorial assumes all source files are relative to the **C:\Mwos** directory.

- 
- Step 1. Start the Hawk IDE from Windows, by selecting *Microware Hawk IDE* from the Programs->Enhanced OS-9 for X86 folder
  - Step 2. When the Hawk IDE appears, go to the Project menu and select Project\_Space->New.
  - Step 3. When the *Create a New Project Space* dialog appears, select *Browse* and navigate to the \mwos\PROJECTS directory. Type *HomeWatch* in the File\_name field and select *Open*. Select *OK* to create the project space.
  - Step 4. Select the *New Component* icon at the top, right of the component window. Type *HomeWatch* in the Name field. Verify that the Default Processor is 80386. Click on the *Next* button.
  - Step 5. There will be only one component in this project. Type *security* in the Name text field. You may also want to add a short description of the program, but no other fields should be changed. click on the *Next* button.

- Step 6. Select the units that will be associated with this component. Navigate to the C:\MWOS\SRC\MAUI\DEMOS\HOMEWATCH folder, and double click on the security.c source file. This is the only source file that is needed for this program. Click on the *Finish* button.
- 

Once these steps are completed, Hawk will generate the dependencies for the source file, and you will be presented with the security component on the left side of the Hawk IDE window.

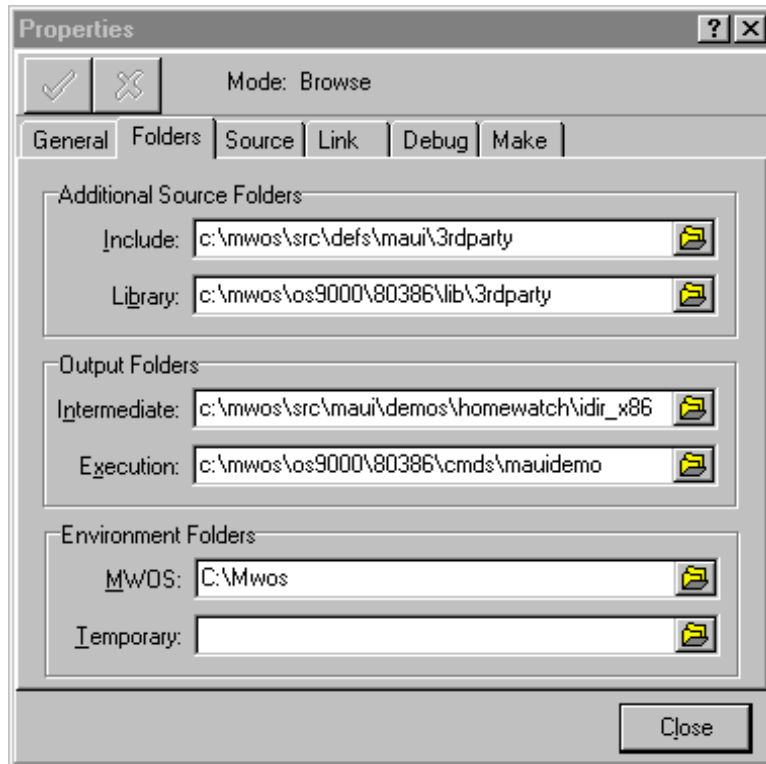
## Customizing the Component

Although the project is now created, we still need to customize the security component. Specifically, we need to tell Hawk about the location of 3rd party header files as well as both Microware and 3rd party libraries used to construct the security application.

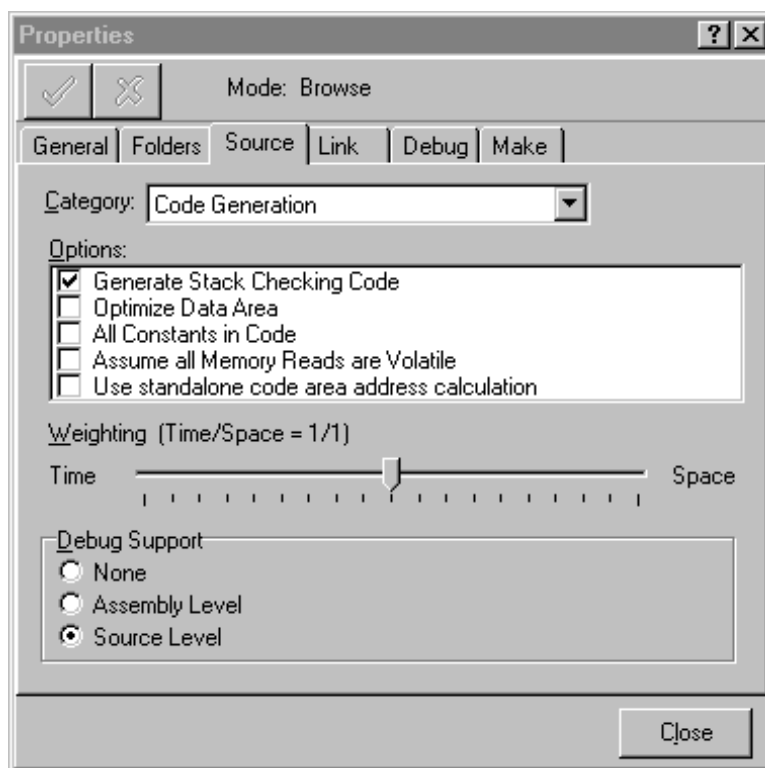
---

- Step 1. From the Project menu, select *Properties*.
- Step 2. When the Properties dialog box appears, select the *Folders* tab. You are presented with 6 text fields.
- In the Include: text field, type:  
C:\MWOS\SRC\DEFS\MAUI\3RDPARTY
  - In the Library: text field, type:  
C:\MWOS\OS9000\80386\LIB\3RDPARTY
  - In the Intermediate: text field, type:  
C:\MWOS\SRC\MAUI\DEMOS\HOMEWATCH\IDIR\_X86
  - In the Execution: text field, type:  
C:\MWOS\OS9000\80386\CMD5\MAUIDEMO





Step 3. Select the *Source* Tab.

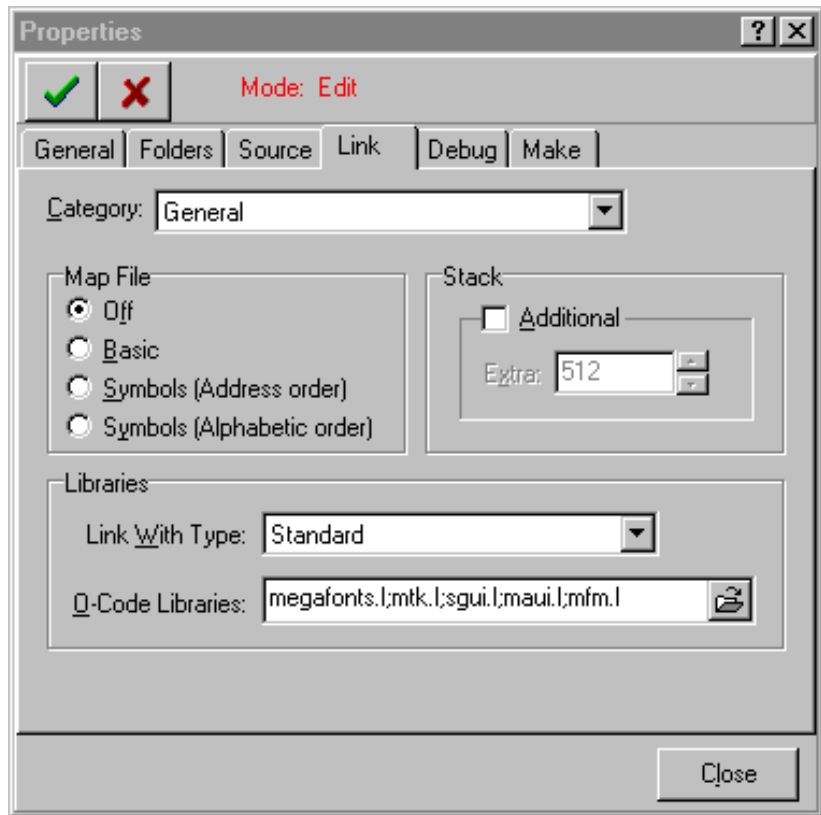


Step 4. In the Category: list box, select *Code Generation*. Notice the area at the bottom of the dialog box labeled Debug Support. Select the option *Source Level*. This allows us the option of source debugging the application if desired.

**Step 5.** Select the *Link* tab.

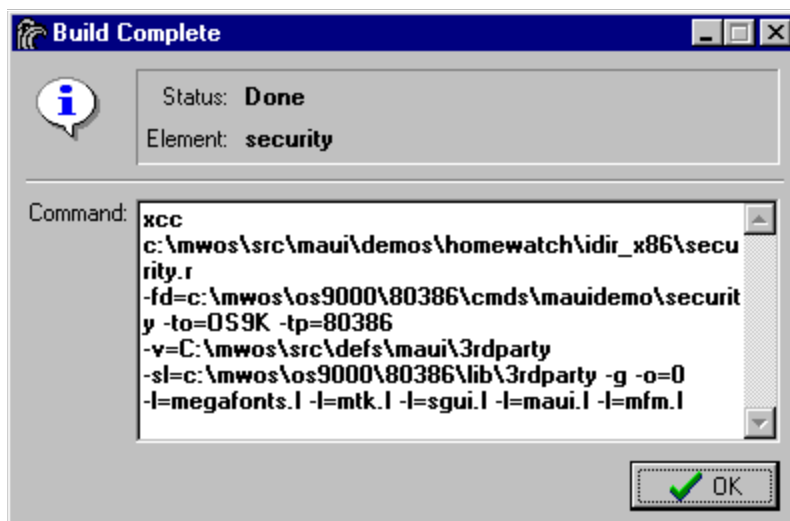
Here we must tell Hawk which libraries to link to specifically. In the O-Code Libraries: text field, type the following:

`megafonts.l;mtk.l;sgui.l;maui.l;mfm.l`

**Step 6.** Click on the *Close* button.

## Building the Project

Now we are ready to build the project. Go to the Project menu and select *Build*. Hawk will display a dialog box showing the progress of the compilation of the security component. If no errors occur, the dialog box will show Status: Done and you can dismiss the box by clicking the *OK* button. If errors do occur, review the above procedures and be sure you followed them exactly as outlined.



## Loading and Executing the HomeWatch Application

To load the application, right click on security in the components box on the left side of the Hawk IDE. When the context box appears, select *Load*. Since this is your first time loading this newly created component, a dialog box will appear asking for the name or IP address of the target to load this component to (os9test in the preceding example). Type this information and click on *OK*. The security module will then be loaded to the target.

An additional file, which contains graphic assets, must also be loaded.

- 
- Step 1. Go the Target menu and select *Load*.
- Step 2. In the new dialog box, navigate to the C:\MWOS\OS9000\80386\CMD5\MAUIDEMO folder, Double click on the file *security\_data*, and then click *Load*.

With both the security and security\_data files loaded, you can run the security demo. For example, from the OS-9 console, run the MAUI input process by typing:

```
maui_inp ^255 <>>>/nil&
```

at the OS-9 command prompt and then running the security program by typing *security* at the command prompt.



---

## Note

The maui\_inp process only needs to be started once. A keyboard must be attached to the target system if *Keyboard Support* was enabled when the OS-9 image was built.

---

## Security

Upon running the security program, you are presented with a graphic screen containing a keypad. The purpose of this program is to secure a known code and exit, possibly allowing a future HomeWatch application to begin running, or to simply turn off an armed home alarm system.

Try typing in a random code. Upon an unsuccessful entry, you will be notified and allowed to input another code. The correct code is 1-2-4-5-5.

## Modifying the Key Code

Changing the length of the key code or even the key code itself is easy. From Hawk, click on security.c on the left hand side of the IDE to edit the source file. Look around line 47 for the following code:

```
#define CODE_LEN 5
static char alarmCode[CODE_LEN] = { '1', '2', '4',
 '5', '5' };
```

The CODE\_LEN definition can be changed to accommodate any code length. Keep in mind that if you change the length, you should also change the proper code in the alarmCode array to match the length.

Once the code has been changed:

- 
- Step 1. Save the file (File->Save)
  - Step 2. Build the security application again by right-clicking on the security component on the left side of the Hawk IDE and selecting *Build*.
  - Step 3. Once the build has properly completed, we need to be sure the old security application in our target is not running. We must unlink the previously loaded security module by going to the Target menu and selecting *Unlink*. A dialog box appears. Replace the name of the target with your target's IP address or name. Fill in the Module: field with the name security (a link count of 1 should be sufficient to unlink the old security module, so leave this field as is) Click on the Unlink button.
  - Step 4. Now load the new security module by right-clicking on the security component like before, and selecting *Load*.
  - Step 5. Run the application again from the telnet session to your target.
-

## Using Cross Hosted Utilities

The following utilities may be executed on the Windows95/98 or NT system to access a OS-9 formatted floppy (RBF). Users may use the cross hosted utilities in much the same way they do from OS-9. Note: The Wizard uses the Cross Hosted Utilities when creating boot media.

### os9dir

The os9dir utility displays a formatted list of file names of the specified directory file on standard output. Refer to the [dir](#) utility in the [Utilities Reference Manual](#) for specific information on command line options and additional functionality.

```
C:\>os9dir -e /d0
```

```

 Directory of /d0 05:27:54
 Owner Last modified Attributes Block Bytecount Name

0.0 98/10/310612d----swr-swr-swr6128CMDS
0.0 98/10/310612d----swr-swr-swr4256SYS
0.0 98/10/310613-----wrF52976firstboot
0.0 98/09/101952-----wr--wr--wrA376iplfd
0.0 98/09/101952-----wr--wr--wrB504iplhd
0.0 98/09/101952-----wr--wr--wrD480iplhdnoq
0.0 98/10/310613-----wr13B36160sysboot
```

### os9dcheck

The os9dcheck utility is a diagnostic tool which detects the condition and general integrity of the directory/file linkages of a disk device. Refer to the [dcheck](#) utility in the [Utilities Reference Manual](#) for specific information on command line options and additional functionality.

```
C:\>os9dcheck /d0
```

```

volume - 'OS9 Boot Disk' on device /d0
$00000b3f total blocks on media
$00000200 total bytes in bitmap
block $00000001 is start of bitmap fd
```

```

block $0000000f is start of low level boot file
block $0000013b is start of bootstrap fd
block $00000002 is start of root directory fd
building allocation map...
checking allocation map...

'OS9 Boot Disk' file structure is intact
3 directories, 8 files, 6 hard links
999424 of 1474048 bytes (0.95 of 1.40 meg) used on media

```

## os9list

The os9list utility displays text lines from the specified path(s) to standard output. Refer to the [list](#) utility in the [Utilities Reference Manual](#) for specific information on command line options and additional functionality.

```

C:\>os9list /d0/sys/startup
-nt
-nx
*
* In case multi-term is running.
*
mshell -lp="OS9_w1: " <>>>/mterm1&
mshell -lp="OS9_w2: " <>>>/mterm2&
mshell -lp="OS9_w3: " <>>>/mterm3&
ex mshell -lp="OS9_w0: " <>>>/term&

```





## Note

The main difference between the use of the OS-9 cross hosted utilities and the OS-9 resident utilities is the lack of wild card support. In OS-9, the shell provides this service. Under Windows95/98 or NT the standard shell does not provide this service. We can however pipe commands to perform the same functionality. Also note that the OS-9 utilities will work on the native Windows95/98 NT file system.

```
C:\MWOS\OS9000\80386\CMD5>os9dir -u p* | ident -qz
p2init size #4880 owner 0.0 ed #15 good crc #CF17B4
padrom size #3904 owner 1.0 ed #6 good crc #837D7A
park size #2736 owner 1.0 ed #3 good crc #33F98D
pcnfsd size #52632 owner 0.0 ed #16 good crc #C03ABA
pd size #5304 owner 1.0 ed #28 good crc #B89E0E
pinfo size #5368 owner 1.0 ed #3 good crc #74314E
```

## Additional Cross Hosted Utilities

```
compare filesos9cmp /d0/a /d0/bos9cmp.exe
dump filesos9dump /d0 or os9dump /d0@os9dump.exe
merge filesos9merge /d0/a >/d0/bos9merge.exe
touch filesos9touch /d0/samos9touch.exe
format mediaos9format /d0os9format.exe
change attributeos9attr -epege /d0/moduleos9attr.exe
make bootableos9bootgen /d0 -i=iplfd -l=corebootos9bootgen.exe
change owneros9chown 1.0 /d0/fileos9chown.exe
copy filesos9copy myfile -w=/d0os9copy.exe
check diskos9dcheck /d0os9dcheck.exe
delete filesos9del /d0/samos9del.exe
show directoryos9dir -e /d0os9dir.exe
show free spaceos9free /d0os9free.exe
list filesos9list /d0/sys/passwordos9list.exe
make directoryos9mkdir /d0/SYS /d0/CMD5os9mkdir.exe
rename filesos9rename /d0/sam /d0/fredos9rename.exe
```



---

**Note**

os9list and os9copy both support conversion options.

\$ os9copy /d0/sam fred -cod ? convert to DOS from OS9

\$ os9copy fred /d0/sam -cdo ? convert to OS-9 from DOS

---

## Supported Devices

---

Following is a list of the supported devices in this release. Please refer to <http://www.microware.com/X86relnotes> for late breaking news and to <http://bbs.microware.com> for driver updates.

### Ethernet Controllers

- **3COM PCI** EtherLink XL
- **3COM PCI** EtherLink III
- **3COM ISA** EtherLink III
- **SMC Ultra 83C790**
- **SMC 8390**
- **3COM PCMCIA**
- **DEC 21140**
- **AM79C961 & AM79C73A**
- **NE2000** PCI/ISA
- **Cirrus Logic CS8900**

### Maui VGA Support

- Generic VGA mode 13 ( 320x200x8bpp )
- Generic VGA mode 12 & "X" ( 640x480x4bpp & 360x480x8bpp )
- Cirrus Alpine Series - CL-GD5434, CL-GD5480 etc. ( up to 1024x768x24bpp )
- VESA ( INT 10h ) driver
- ISA banked

### Sequential Device Support

- **VGA Graphics / Keyboard**
- **Serial Mouse**
- **PS2 Mouse**
- **16550 Serial**

- **Digiboard**
- **HostessI**
- **Risicom**
- **Parallel Printer**

### **Physical Disk Media**

- **IDE Standard**
- **PCMCIA IDE**
- **IDE Standard**
- **DiskOnChip**
- **DiskOnChip Descriptors**
- **PC AT Style Floppy**
- **Floppy Descriptors**
- **Symbios 810,810A,825,825A and 875 PCI SCSI controllers—Wide, Ultra and Ultra Wide**
- **Diamond FirePort20 and FirePort40—Wide, Ultra and Ultra Wide**
- **Adaptec 1540/1542 ISA**
- **Adaptec 2940, 2940U and 2940UW**
- **SCSI Descriptors**

### **System Devices**

- **Real Time Clock**

### **Additional Devices**

- **PPP and SLIP**

## **Ethernet Controllers**

Please refer to <http://www.microware.com/X86relnotes> and to <http://bbs.microware.com> for information on new drivers.



---

**Note**

Some Network Interface Cards require that a setup disk, included with the card, is ran before the card is installed in a system running OS-9.

The setup disk is required for configuring the connection type for cards which support multiple interfaces, such as connections for 10Base-T, 10Base-2 or AUI. The setup disk may also be needed to configure the card for a specific interrupt or I/O address.

---

**3COM PCI**

3C900B-TPO - 10Base-T TPO NIC

3C900B-CMB - 10Base-T/10Base-2/AUI Combo

3C905-T4 - 10/100 Base-T4 (RJ-45) - 3C905-T4 Fast Etherlink XL

3C905B-TX - 10/100Base-TX NIC

3CSOHO100-TX - 10/100 Base-TX NIC - Office Connect 10/100

3C900-TPO - 10Base-T TPO NIC

System State Debugging - Supported

**Default Settings**

PORTADDR     NA

IRQVECTOR    NA

CONNTYPE     INF\_EXT /\* Auto \*/

**Solving Configuration Issues****Connection Type**

The default connection type is set to INF\_EXT (auto). For the 3COM ISA card, this implies the card setup program has been used and has setup the card connection type. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following:

1. Use the 3COM setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.
3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

```
(Super)[/h0/sys/>] netstat -in
```

```
NameMtu NetworkAddressIpktsIerrsOpktsOerrsColl
lo0 1536<Link>0000 0
lo0 1536127127.0.0.100000
enet01500<Link>00.00.C0.91.4F.96551103500
enet01500182.52.109182.52.109.2555103500
```

## How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf\_desc.h, looking in the "#ifdef spe30\_pci" section for CONNTYPE, which you should set to the appropriate value from the following list:

INF\_AUI = AUI Connection type

INF\_BNC = BNC connection type

INF\_UPT = 10BaseT (RJ45)

INF\_EXT = Use same connection type determined in 3COM setup program

```
/*
 * From spf_desc.h
 */

/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */

#define CONNTYPEINF_EXT
```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509 directory and type "os9make -f=spfdesc.mak".

Next time you run the Wizard, it will use the new descriptor.

## Boomerang

The source code for the spe509 driver includes “#if defined(BOOMERANG)” sections to allow only including support for the newer 3COM PCI based cards. Each card is now defined in a constant table and as such the driver makefile used must be modified to include both the define for “BOOMERANG” and the compiler option “-c” to force constant code data.

```
/* spfdrvr.mak - add the following define and compiler option */

DEFINES = -c -dBOOMERANG

/* spfdesc.mak - add the following define */
MACROS = -dBOOMERANG
```

## DMA

To allow support for the newer 3COM “B” based cards, DMA support with ring buffers has been added. The size of the ring buffers may be set in the “spf\_desc.h” file.

```
#define RX_RING_CNT32/* Number of buffers in BOOMERANG recv ring */
#define TX_RING_CNT16/* Number of buffers in BOOMERANG xmit ring */
```

## Time-out Options

To allow support with switches and slow hubs the time-out for checking for link beat has been increased. This change effects 3COM NON-B parts as well as PCMCIA CARDS using UTP connections. The default time-out prior to this change was 750ms. Most switches take two to three seconds to sync. A loop count has been added.

```
/*
 * When a connection type is tried we will wait for the time
 * specified in LINK_BEAT_ITER and LINK_BEAT_SLEEP_TIME.
 * This should address the problem with not being able to work
 * with switches. Most switches will take 2 to 3 seconds, we will wait up to
 * 5.25 seconds (192/256ths)*7.
 *
 */

#define LINK_BEAT_ITER 7
#define LINK_BEAT_SLEEP_TIME 0x800000c0 /* 192/256ths of a second (750 ms) */
```



## Note

The **PCIV** utility may be used to examine a network card. This utility displays vendor and device ID's for each installed PCI device.

To find out if your card has been tested with OS-9, run the `pciv` command and look at the vendor and device ID's. The vendor ID should be 0x10B7 for all 3COM network cards. Network cards with the following device ID's have been tested with OS-9 drivers shipping with this release.

|                   |        |     |
|-------------------|--------|-----|
| 3COM 3C509        | 0x5900 |     |
| 3COM 3C900-TPO    | 0x9000 |     |
| 3COM 3C900        | 0x9001 |     |
| 3COM 3C900B-TPO   | 0x9004 |     |
| 3COM 3C900B-CMB   | 0x9005 |     |
| 3COM 3C905-T4     | 0x9051 | (2) |
| 3COM 3C905B-TX    | 0x9055 | (1) |
| 3COM 3CSOHO100-TX | 0x7646 | (1) |

Support for the following cards is included with the driver, however, these cards were not tested prior to the release.

|                            |        |
|----------------------------|--------|
| 3COM 3C905-TX              | 0x9050 |
| 3COM 10/100 COMBO Deluxe   | 0x9058 |
| 3COM 10Base-T/10Base-2/TPC | 0x9006 |
| 3COM 10Base-FL NIC         | 0x900A |
| 3COM 100Base-FX NIC        | 0x905A |



|                                    |        |
|------------------------------------|--------|
| 3COM Tornado NIC                   | 0x9200 |
| 3COM 10/100 Base-TX NIC (Python-H) | 0x9800 |
| 3COM 10/100 Base-TX NIC (Python-T) | 0x9805 |

### **Additional Notes**

- 1 100BaseT support is included for the 3C905B-TX and 3CSOHO100-TX.
  - 2 The 3C905-T4 has been tested with 10BaseT only.
- 

## **3COM ISA**

3COM ISA EtherLink III

**System State Debugging** - Supported

### **Default Settings**

```
PORTADDR 0x340 /* IO port for ISA */
IRQVECTOR 0x43 /* IRQ vector */
CONNTYPE INF_EXT /* Auto */
```

## **Solving Configuration Issues**

### **Connection Type**

The default connection type is set to INF\_EXT (auto). For the 3COM ISA card, this implies the card setup program has been used and has setup the card connection type. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following.

1. Use the 3COM setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.

3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

### Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ3. In this case you have the following options.

1. Disable the COM2 serial port from the BIOS to allow IRQ3 to function with this card.
2. Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

If it seems like we should be getting interrupts this can be tested.

Use the command `irqs` to see a list of interrupts, e.g.:

```
(Super)[/h0/sys/>] irqs
```

```
PC-AT Compatible 80386 OS9 For Embedded Systems
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver  | dev list |
|--------|--------|-------|------------|------------|---------|----------|
| 7      | (\$07) | 10    | \$0003c444 | \$0010f7b4 | fpu     | <na>     |
| 14     | (\$0e) | 1     | \$0003c3a4 | \$00110113 | vectors | <na>     |
| 64     | (\$40) | 10    | \$00ff40b0 | \$0011098f | tk8253  | <na>     |
| 65     | (\$41) | 10    | \$00ffa680 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e85db0 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e84a40 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e82980 | \$00120582 | sc8042m | <na>     |
| 74     | (\$4a) | 1     | \$00ff02d0 | \$001f9504 | spe509  | <na>     |
| 78     | (\$4e) | 10    | \$00ff4f30 | \$00137906 | rb1003  | <na>     |

In this case, we can go into RomBug by typing `break` and placing a breakpoint at the ISR.

```
$ break
RomBug: b 1f9504
RomBug: g
```

and then pinging a machine on the net:

```
$ ping 182.52.109.13
```

( using the actual address of another machine on the network, rather than the one shown above).

If interrupts are running you should be presented a Rombug prompt at the breakpoint address. You can type g to see if you get another interrupt or k to kill the breakpoint.

### Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case, the OS-9 command netstat -in will not show the card as available.

The following netstat example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

### How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf\_desc.h  
and look for the #ifdef spe30\_isa section.

Change the fields below as required.

INF\_AUI = AUI Connection type

INF\_BNC = BNC connection type

INF\_UPT = 10BaseT (RJ45)

INF\_EXT = Use same connection type determined in 3COM setup program

```
/*
 * From spf_desc.h
 */

#define PORTADDR0x340/* IO port for ISA*/
#define IRQVECTOR0x43/* IRQ vector */
```

```
/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */
#define CONNTYPEINF_EXT
```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509 directory and type:

```
C:> os9make -f=spfdesc.mak
```

Next time you run the Wizard the new descriptor will be used.

### Low-level system changes

If system state debugging is used, you must change the low level system by modifying the following lines from the file

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des:

```
#define LLE509_PORT_ADDRESS 0x340
#define LLE509_IF_VECTOR 0x43
```

as required by the system. For example, for IRQ10, here are the changes required.

```
#define LLE509_PORT_ADDRESS 0x340
#define LLE509_IF_VECTOR 0x4a
```

The Wizard will automatically re-make the cnfgdata module.

## SMC Ultra 83C790

### System State Debugging - Supported

#### Default Settings

|          |            |                                |
|----------|------------|--------------------------------|
| PORTADDR | 0x00000300 | /* Base address of hardware */ |
| VECTOR   | 0x4a       | /* Port vector */              |
| CPU_BASE | 0xcc000    | /* CPU BASE */                 |
| RAM_SIZE | 0x400      | /* RAM Size */                 |

### Solving Configuration Issues

#### Connection Type

On the SMC 83C790, the hardware connection type is setup in hardware on the board. Refer to the SMC 83C70 documentation that came with the card used.

## Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ10. In this case you have the following options.

1. Resolve the conflict by moving the device using IRQ10.
2. Choose an interrupt that matches the system configuration such as IRQ15 (0x4f). In this case the OS-9 device descriptor must be changed and the board jumpers must be changed to set the interrupt to be used.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

Use the command `irqs` to see a list of interrupts.

```
(Super)[/h0/sys/>] irqs
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver   | dev list |
|--------|--------|-------|------------|------------|----------|----------|
| 7      | (\$07) | 10    | \$0003a404 | \$0010f844 | fpu      | <na>     |
| 14     | (\$0e) | 1     | \$0003a364 | \$00112027 | vectors  | <na>     |
| 64     | (\$40) | 10    | \$004f2850 | \$00112633 | tk8253   | <na>     |
| 65     | (\$41) | 10    | \$004fa320 | \$00122226 | sc8042m  | <na>     |
| 74     | (\$4a) | 5     | \$004f0030 | \$001fed47 | sp83c790 | <na>     |

In the case above we can go into RomBug by typing `break` and placing a break at the ISR.

```
$ break
RomBug: b 1fed47
RomBug: g
```

and then pinging a machine on the net:

```
$ ping 182.52.109.13
```

( using the actual address of another machine on the network, rather than the one shown above).

If interrupts are running you should be presented a Rombug prompt at the breakpoint address. You can type `g` to see if you get another interrupt or `k` to kill the breakpoint.

## Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case the OS-9 command `netstat -in` will not show the card as available.

The following `netstat` example shows a working network card configured with IP address 182.25.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

**Note:** On the SMC 83C790, the ring buffers are located in the 384k hole (1MB-384K - 1MB).

The default location of 0xcc000 may be used by another system device. You can dump the memory at this location to see if it looks like packet data. The OS-9 command `dump` may be used to dump the memory to see if it looks correct.

### Good case

```
(Super)[/h0/sys/>] dump -a 0xcc000
```

| Addr     | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8          | 9            | A    | B    | C | D | E | F | 0 | 2 | 4 | 6 | 8 | A | C | E |
|----------|------|------|------|------|------|------|------|------|------------|--------------|------|------|---|---|---|---|---|---|---|---|---|---|---|---|
| 000cc000 | 0000 | c040 | bbc7 | 0000 | c091 | 4f96 | 0800 | 4500 | ..@;       | G..@.O...E.  |      |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc010 | 002a | 037b | 0000 | 4006 | 30c4 | b634 | 6d19 | b634 | .*.{       | ..@.0D64m.64 |      |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc020 | 6d0d | 0017 | 0a20 | 0121 | 71fd | 07fa | 776a | 5018 | m....      | !q}.zwjP.    |      |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc030 | 4000 | 7840 | 0000 | 0d0a | 2020 | 4164 | 6472 | 2020 | @.x@       | .... Addr    |      |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc040 | 2020 | 2030 | 2031 | 2020 | 3220 | 3320 | 2034 | 2035 | 0          | 1            | 2    | 3    | 4 | 5 |   |   |   |   |   |   |   |   |   |   |
| 000cc050 | 2020 | 3620 | 3720 | 2038 | 2039 | 2020 | 4120 | 4220 | 6          | 7            | 8    | 9    | A | B |   |   |   |   |   |   |   |   |   |   |
| 000cc060 | 2043 | 2044 | 2020 | 4520 | 4620 | 3020 | 3220 | 3420 | C          | D            | E    | F    | 0 | 2 | 4 |   |   |   |   |   |   |   |   |   |
| 000cc070 | 3044 | 3634 | 6d2e | 3634 | 0d0a | 3030 | 3063 | 6330 | 0D64m.64.. | 000cc0       |      |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc080 | 3230 | 2020 | 3664 | 3064 | 2030 | 3031 | 3720 | 3061 | 20         | 6d0d         | 0017 | 0a   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc090 | 3230 | 2030 | 3132 | 3120 | 3731 | 6664 | 2030 | 3766 | 20         | 0121         | 71fd | 07f  |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0a0 | 6120 | 3737 | 3661 | 2035 | 3031 | 3820 | 6d2e | 2e2e | a          | 776a         | 5018 | m... |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0b0 | 2e20 | 2e21 | 717d | 2e7a | 776a | 502e | 0d0a | 3030 | .          | !q}.zwjP...  | 00   |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0c0 | 3063 | 6330 | 3330 | 2020 | 3430 | 3030 | 2037 | 3834 | 0cc030     | 4000         | 784  |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0d0 | 3020 | 3331 | 3332 | 2033 | 3132 | 3020 | 3337 | 3331 | 0          | 3132         | 3120 | 3731 |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0e0 | 2036 | 3636 | 3420 | 3230 | 3330 | 2033 | 3736 | 3620 | 6664       | 2030         | 3766 |      |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0f0 | 3230 | 2030 | 3132 | 3120 | 3731 | 6664 | 2030 | 3766 | 20         | 0121         | 71fd | 07f  |   |   |   |   |   |   |   |   |   |   |   |   |

### Bad case - VGA BIOS at location desired

```
(Super)[/h0/sys/>] dump -a 0xe0000
```

| Addr  | 0     | 1     | 2     | 3     | 4     | 5     | 6     | 7     | 8     | 9     | A     | B     | C     | D     | E     | F     | 0     | 2     | 4     | 6     | 8     | A     | C     | E     |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- | ----- |

```

000e0000 55aa 40e9 ce00 4c43 3c00 0000 77cc 5649 U*@iN.LC<...wLVI
000e0010 4445 4f20 0000 0000 0062 40e9 2840 4942 DEOb@i(@IB
000e0020 4d20 434f 4d50 4154 4942 4c45 2e2a 0000 M COMPATIBLE.*..
000e0030 f7c8 30df 7a0c 030d aa0c 330d 4744 3533 wh0_z...*3.GD53
000e0040 3230 2056 4741 2042 494f 5320 5665 7273 20 VGA BIOS Vers
000e0050 696f 6e20 332e 3132 2020 2020 2020 200d ion 3.12 .
000e0060 0a43 6f70 7972 6967 6874 2028 6329 2043 .Copyright (c) C
000e0070 6972 7275 7320 4c6f 6769 6320 496e 632e irrur Logic Inc.
000e0080 2031 3938 372d 3139 3931 2e0d 0a43 6f70 1987-1991...Cop
000e0090 7972 6967 6874 2028 6329 2041 7761 7264 yright (c) Award
000e00a0 2053 6f66 7477 6172 6520 496e 632e 2031 Software Inc. 1
000e00b0 3938 342d 3139 3838 2e20 416c 6c20 5269 984-1988. All Ri
000e00c0 6768 7473 2052 6573 6572 7665 642e 0d0a ghts Reserved...
000e00d0 0a00 9043 55e8 947e 33c0 8ed8 bae8 46b8 ...CUh.~3@.X:hF8
000e00e0 1600 ef52 ba02 018a c6ef 5a48 efe8 2610 ..oR:...FoZHoh&.
000e00f0 bae8 4a33 c0ef b001 e85d 45c7 0640 0062 :hJ3@o0.h]EG.@.b

```

## How to modify the OS-9 descriptor

Edit the file MWOS/OS9000/80386/PORTS/PCAT/SPF/SP83C790/DEFS/spf\_desc.h

and change the following fields as needed:

```

#define PORTADDR0x00000300/* Base address of hardware */
#define VECTOR0x4a/* Port vector */
#define CPU_BASE0xcc000/* CPU BASE */
#define RAM_SIZE0x4000/* RAM Size */

```

Then, remake the descriptor: change to the directory MWOS/OS9000/80386/PORTS/PCAT/SPF/SP83C790 and type "os9make -f=spfdesc.mak"

Next time you run the Wizard, it will use the new descriptor.

## Low-level System Changes

We must now change the low level system if System State debugging is used.

In the file MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des find the following lines:

```

#define LL83790_PORT_ADDRESS0x300
#define LL83790_IF_VECTOR0x4a

```

These determine the port addresses and/or IRQ information, and should be changed as required by the system. For IRQ15 here are the changes required:

```

#define LL83790_PORT_ADDRESS0x300
#define LL83790_IF_VECTOR0x4f

```

The Wizard will automatically re-make the cnfgdata module.

**Note**

The SMC 83C795 part is not yet supported.

**SMC 8390****System State Debugging - Not Supported****Default Settings**

**PORTADDR0x00000300/\* Base address of hardware \*/**

VECTOR 0x4a /\* Port vector \*/

CPU\_BASE 0xcc000 /\* CPU BASE \*/

RAM\_SIZE 0x4000 /\* RAM Size \*/

**Solving Configuration Issues****Connection Type**

On the SMC 8390 the hardware connection type is setup in hardware on the board. Refer to the SMC 83C70 documentation that came with the card used.

**Interrupt Conflict**

Another problem may be the interrupt used. The default interrupt is IRQ10. In this case you have the following options.

Resolve the conflict by moving the device using IRQ10.

3. Choose a interrupt that matches the system configuration such as IRQ15 (0x4f). In this case the OS-9 device descriptor must be changed. In this case the board jumpers must be changed to set the interrupt to be used.
4. If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.



5. Use the command `irqs` to see a list of interrupts.
6. (Super)[/h0/sys/>] `irqs`
7. PC-AT Compatible 80386 OS9 For Embedded Systems (X86) V1.0
8. vector (\$) prior drivstat irq svc driver dev list

|    |        |    |            |            |         |      |
|----|--------|----|------------|------------|---------|------|
| 7  | (\$07) | 10 | \$0003a404 | \$0010f844 | fpu     | <na> |
| 14 | (\$0e) | 1  | \$0003a364 | \$00112027 | vectors | <na> |
| 64 | (\$40) | 10 | \$004f2850 | \$00112633 | tk8253  | <na> |
| 65 | (\$41) | 10 | \$004fa320 | \$00122226 | sc8042m | <na> |
| 74 | (\$4a) | 5  | \$004f0030 | \$001fed47 | sp8390  | <na> |

In the case above we can go into RomBug by typing `break` and placing a break at the ISR.

**\$ break**

```
RomBug: b 1fed47
RomBug: g
```

and then pinging a machine on the network.

```
$ ping 182.52.109.13
```

(using the actual address of another machine on the network, rather than the one shown here)

If interrupts are running you should be presented a *Rombug* prompt at the breakpoint address. You can type `g` to see if you get another interrupt or `k` to kill the breakpoint.

## Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case the OS-9 command `netstat -in` will not show the card as available.

The following `netstat` example shows a working network card configured with IP address 12.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

**Note:** On the SMC 8390 the ring buffers are located in the 384k hole (1MB-384K - 1MB).

The default location of 0xcc000 may be used by another system device. You can dump the memory at this location to see if it looks like packet data. The OS-9 command dump may be used to dump the memory to see if it looks correct.

### Good case

```
(Super)[/h0/sys/>] dump -a 0xcc000
```

| Addr     | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8                | 9 | A | B | C | D | E | F | 0 | 2 | 4 | 6 | 8 | A | C | E |
|----------|------|------|------|------|------|------|------|------|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000cc000 | 0000 | c040 | bbc7 | 0000 | c091 | 4f96 | 0800 | 4500 | ..@@;G..@.O...E. |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc010 | 002a | 037b | 0000 | 4006 | 30c4 | b634 | 6d19 | b634 | .*.{..@.0D64m.64 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc020 | 6d0d | 0017 | 0a20 | 0121 | 71fd | 07fa | 776a | 5018 | m.... !q}.zwjP.  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc030 | 4000 | 7840 | 0000 | 0d0a | 2020 | 4164 | 6472 | 2020 | @.x@.... Addr    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc040 | 2020 | 2030 | 2031 | 2020 | 3220 | 3320 | 2034 | 2035 | 0 1 2 3 4 5      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc050 | 2020 | 3620 | 3720 | 2038 | 2039 | 2020 | 4120 | 4220 | 6 7 8 9 A B      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc060 | 2043 | 2044 | 2020 | 4520 | 4620 | 3020 | 3220 | 3420 | C D E F 0 2 4    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc070 | 3044 | 3634 | 6d2e | 3634 | 0d0a | 3030 | 3063 | 6330 | 0D64m.64..000cc0 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc080 | 3230 | 2020 | 3664 | 3064 | 2030 | 3031 | 3720 | 3061 | 20 6d0d 0017 0a  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc090 | 3230 | 2030 | 3132 | 3120 | 3731 | 6664 | 2030 | 3766 | 20 0121 71fd 07f |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0a0 | 6120 | 3737 | 3661 | 2035 | 3031 | 3820 | 6d2e | 2e2e | a 776a 5018 m... |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0b0 | 2e20 | 2e21 | 717d | 2e7a | 776a | 502e | 0d0a | 3030 | . !q}.zwjP...00  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0c0 | 3063 | 6330 | 3330 | 2020 | 3430 | 3030 | 2037 | 3834 | 0cc030 4000 784  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0d0 | 3020 | 3331 | 3332 | 2033 | 3132 | 3020 | 3337 | 3331 | 0 3132 3120 3731 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0e0 | 2036 | 3636 | 3420 | 3230 | 3330 | 2033 | 3736 | 3620 | 6664 2030 3766   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000cc0f0 | 3230 | 2030 | 3132 | 3120 | 3731 | 6664 | 2030 | 3766 | 20 0121 71fd 07f |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

### Bad case - VGA BIOS at location desired

```
(Super)[/h0/sys/>] dump -a 0xe0000
```

| Addr     | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | 8                | 9 | A | B | C | D | E | F | 0 | 2 | 4 | 6 | 8 | A | C | E |
|----------|------|------|------|------|------|------|------|------|------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000e0000 | 55aa | 40e9 | ce00 | 4c43 | 3c00 | 0000 | 77cc | 5649 | U*@iN.LC<...wLVI |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0010 | 4445 | 4f20 | 0000 | 0000 | 0062 | 40e9 | 2840 | 4942 | DEO .....b@i(@IB |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0020 | 4d20 | 434f | 4d50 | 4154 | 4942 | 4c45 | 2e2a | 0000 | M COMPATIBLE.*.. |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0030 | f7c8 | 30df | 7a0c | 030d | aa0c | 330d | 4744 | 3533 | wH0_z...*.3.GD53 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0040 | 3230 | 2056 | 4741 | 2042 | 494f | 5320 | 5665 | 7273 | 20 VGA BIOS Vers |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0050 | 696f | 6e20 | 332e | 3132 | 2020 | 2020 | 2020 | 200d | ion 3.12 .       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0060 | 0a43 | 6f70 | 7972 | 6967 | 6874 | 2028 | 6329 | 2043 | .Copyright (c) C |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0070 | 6972 | 7275 | 7320 | 4c6f | 6769 | 6320 | 496e | 632e | irrus Logic Inc. |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0080 | 2031 | 3938 | 372d | 3139 | 3931 | 2e0d | 0a43 | 6f70 | 1987-1991...Cop  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e0090 | 7972 | 6967 | 6874 | 2028 | 6329 | 2041 | 7761 | 7264 | yright (c) Award |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00a0 | 2053 | 6f66 | 7477 | 6172 | 6520 | 496e | 632e | 2031 | Software Inc. 1  |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00b0 | 3938 | 342d | 3139 | 3838 | 2e20 | 416c | 6c20 | 5269 | 984-1988. All Ri |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00c0 | 6768 | 7473 | 2052 | 6573 | 6572 | 7665 | 642e | 0d0a | ghts Reserved... |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00d0 | 0a00 | 9043 | 55e8 | 947e | 33c0 | 8ed8 | bae8 | 46b8 | ...CUh.~3@.X:hF8 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00e0 | 1600 | ef52 | ba02 | 018a | c6ef | 5a48 | efe8 | 2610 | ..oR:...FoZHoh&. |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 000e00f0 | bae8 | 4a33 | c0ef | b001 | e85d | 45c7 | 0640 | 0062 | :hJ3@o0.h]EG.@.b |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

## How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/SP8390/DEFS/spf\_desc.h  
and change the following fields as required:

```
#define PORTADDR0x00000300/* Base address of hardware */
#define VECTOR0x4a/* Port vector */
#define CPU_BASE0xcc000/* CPU BASE */
#define RAM_SIZE0x4000/* RAM Size */
```

Then remake the descriptor: change to the  
MWOS/OS9000/80386/PORTS/PCAT/SPF/SP8390 directory and type:

```
os9make -f=spfdesc.mak
```

You have now created a new descriptor. Next time you run the Wizard, it will use the new descriptor.

## 3COM PCMCIA

3COM EtherLink III PC CARD

3COM Megahertz LAN (3CCE589ET) - 10 Mbps LAN PC Card

**System State Debugging** - Supported

### Default Settings

|           |         |                       |
|-----------|---------|-----------------------|
| PORTADDR  | 0x340   | /* IO port for ISA */ |
| IRQVECTOR | 0x43    | /* IRQ vector */      |
| CONNTYPE  | INF_EXT | /* Auto */            |

## Solving Configuration Issues

### Connection Type

The default connection type is set to INF\_EXT (auto). For the 3COM PCMCIA card this implies the card will detect the connection type used. If desired the connection type may be forced. To force the connection type the descriptor must be changed.

### Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ3. In this case you have the following options.

1. Disable the COM2 serial port from the BIOS to allow IRQ3 to function with this card.
2. Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed. Also the PCMCIA socket services setup must be changed to assign the new interrupt to the PCMCIA Ethernet Card.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

Use the command `irqs` to see a list of interrupts.

```
(Super)[/h0/sys/>] irqs
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver  | dev list |
|--------|--------|-------|------------|------------|---------|----------|
| 7      | (\$07) | 10    | \$0003c444 | \$0010f7b4 | fpu     | <na>     |
| 14     | (\$0e) | 1     | \$0003c3a4 | \$00110113 | vectors | <na>     |
| 64     | (\$40) | 10    | \$00ff40b0 | \$0011098f | tk8253  | <na>     |
| 65     | (\$41) | 10    | \$00ffa680 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e85db0 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e84a40 | \$00120582 | sc8042m | <na>     |
| 65     | (\$41) | 10    | \$00e82980 | \$00120582 | sc8042m | <na>     |
| 74     | (\$4a) | 1     | \$00ff02d0 | \$001f9504 | spe509  | <na>     |
| 78     | (\$4e) | 10    | \$00ff4f30 | \$00137906 | rb1003  | <na>     |

In the case above we can go into RomBug by typing `break` and placing a break at the ISR.

```
$ break
```

```
RomBug: b 1f9504
```

```
RomBug: g
```

and then ping a machine on the net.

```
$ ping 182.52.109.13
```

(Using the actual address of another machine on the network, rather than the one shown above.)

If interrupts are running you should be presented a *Rombug* prompt at the breakpoint address. You can type `g` to see if you get another interrupt or `k` to kill the breakpoint.

## Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case the OS-9 command *netstat -in* will not show the card as available.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

## How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509/DEFS/spf\_desc.h

Look for the `#ifdef spe30_isa` section and change the `PORTADDR`, `IRQVECTOR`, and `CONNTYPE` as required.

The permissible values for `CONNTYPE` are:

`INF_AUI` = AUI Connection type

`INF_BNC` = BNC connection type

`INF_UPT` = 10BaseT (RJ45)

`INF_EXT` = Probe connection type

```
/*
 * From spf_desc.h
 */

#define PORTADDR0x340/* IO port for ISA*/
#define IRQVECTOR0x43/* IRQ vector */
/* options for CONNTYPE: INF_AUI, INF_BNC, INF_UTP, INF_EXT (auto) */
#define CONNTYPEINF_EXT
```

Finally, remake the descriptor by changing to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SPE509 directory and typing

```
os9make -f=spfdesc.mak.
```

## Low-level system changes

System state debugging requires a change to the low level system, as well as the PCMCIA socket services information. This is controlled by the contents of the file

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des.

Find the following lines:

```
#define LLE509_PORT_ADDRESS0x340
#define LLE509_IF_VECTOR0x43
#define ETH_CIS_PARAMS"3com=0x340,3"
```

The above port addresses and/or IRQ information should be changed as required by the system. For IRQ 10, here are the changes required:

```
#define LLE509_PORT_ADDRESS0x340
#define LLE509_IF_VECTOR0x4a
#define ETH_CIS_PARAMS"3com=0x340,10"
```

The Wizard will automatically re-make the cnfgdata module.

## DEC 21140

**System State Debugging - Not Supported**

### Cards Supported

SMC EtherPower 10/100 - SMC9332DST

SMC EtherPower 10/100 - SMC9334BDT/SC (Dual)

Intra Server DE504-BA (Quad)

Asante' Fast 10/100

D-Link DFE-500TX ProFast 10/100 Adapter

### Default Settings

PORTADDR     NA

IRQVECTOR    NA

CONNTYPE     INF\_UTP

### Solving Configuration Issues

#### Connection Type

The default connection type is set to INF\_AUI. For the SMC EtherPower 10/100 card this actually is the RJ45 connector. If you are unable to communicate with this card and *netstat -in* shows the device, the connection type may be incorrect. To correct it, you must change the OS-9 device descriptor field that indicates the type of connection in use.

The following netstat example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

## How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/SP21140/DEFS/spf\_desc.h, changing the definition of CONNTYPE as required. The possible values for CONNTYPE are:

```
*
* From spf_desc.h
*/
/*
* Interface/connection type
* Common values:
*
* INF_UTP == MII_10MB == 10Mb/s 21140
* INF_AUI == SRL_10MB == Conventional 10Mb/s 21140
* INF_UTP100 == MII_100MBTX == MII 100Mb/s 21140
* INF_FX100 == MII_100MBFX == MII 100Mb/s 21140
* INF_MII10 == MII_10MB == 10Mb/s 21140
* INF_MII100 == MII_100MB == MII 100Mb/s 21140
*
* Note: Not all common values will work. Below are common
* values used for different cards supported. Much work at
* driver level still remains to allow auto and NWay support.
* Support for DEC21143 may be added in the future once the
* NWay support is added.
*
* SMC EtherPower 10/100 - SMC9332DST
* 10BaseT = INF_AUI
*
* SMC EtherPower 10/100 - SMC9334BDT/SC (Dual)
* 10BaseT = INF_UTP
* 100BaseT = INF_MII100
```

```

*
* Intra Server DE504-BA (Quad)
* 10BaseT = INF_UTP (note: preliminary release support for 21143. No
100BaseT support)
*
* Asante' Fast 10/100
* 10BaseT = INF_UTP
* 100BaseT = INF_MII100
*
* D-Link DFE-500TX ProFast 10/100 Adapter
*
* 10BaseT = INF_UTP
* 10BaseT = INF_MII10
* 100BaseT = INF_MII100
*
*/

#define CONNTYPE INF_UTP

```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SP21140 directory and type:

```
os9make -f=spfdesc.mak
```

You have now created a new descriptor. Next time you run the Wizard, it will use the new descriptor.

### **Adding support for dual and quad channel cards with the Microware Wizard**

The descriptors for the additional Ethernet ports must be added. Edit the spf.ml file in the

MWOS/OS9000/80386/PORTS/PCAT/BOOTS/INSTALL/PORTBOOT directory. Find the entry for spde0. Add spde1 for a dual card or spde1, spde2 and spde3 for a quad card.

Next edit the pcat.ini file located in MWOS/OS9000/80386/PORTS/PCAT/BOOTS/INSTALL/INI directory. Look for the ETHER\_OPTION\_ string and add the entries as required. You must specify the Ethernet information for all extra Ethernet ports used.

The following example adds the three extra Ethernet ports for a quad card.

```
ETHER_OPTION_2=enet1 address 112.16.1.237 broadcast
112.16.255.255 netmask 255.255.000.000 binding /spde1/enet
```



```
ETHER_OPTION_3=enet2 address 122.16.1.237 broadcast
122.16.255.255 netmask 255.255.000.000 binding /spde2/enet
```

```
ETHER_OPTION_4=enet3 address 132.16.1.237 broadcast
132.16.255.255 netmask 255.255.000.000 binding /spde3/enet
```

Once the boot image is created you may boot OS-9 and use "netstat" to see that all cards are active and ready for use. You should see entries for enet0, enet1, enet2 and enet3 if you are using a quad card.

## AM79C961 & AM79C73A

### System State Debugging - Not Supported

#### Default Settings

PORTADDR            0x300

IRQVECTOR           NA

CONNTYPE            NA

#### Solving Configuration Issues

The AM79C961A driver is designed to work in systems where DMA BUS MASTER mode is employed with respect to the AM79C961 or AM79C973 interfaces.

The AM79C961A driver is PLUG & PLAY. Only the base address should be defined to allow multiple card usage.

#### How to modify the OS-9 descriptor

1. Edit the file.  
MWOS/OS9000/80386/PORTS/PCAT/SPF/SP79C961/DEFS/spf\_desc.h, changing the line defining PORTADDR, which reads #define PORTADDR 0x300 /\* Base address of hardware \*/, to give PORTADDR the desired value.
2. Next re-make the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/SP79C961 directory and type the command `os9make -f=spfdesc.mak`

You have now created a new descriptor. The next time you run the Wizard, it will use the new descriptor.

## NE2000

ZF NetDisplay

ACCTON - EN166X MPX 2 Ethernet

D-LINK DE-220PCT - 10Mbps Combo 16-Bit Ethernet ISA Adapter

Compex - ReadyLink 2000 - PCI 32-bit

### System State Debugging - Supported

#### Default Settings

PORTADDR 0x340 /\* IO port for ISA \*/

IRQVECTOR 0x49 /\* IRQ vector \*/

CONNTYPE INF\_EXT /\* Auto \*/

#### Board Setup Issues

##### ZF NetDisplay

use <CDROM>:\Drivers\Ethernet\Realtek\RSET8019.EXE"

to determine the IO address and IRQ required.

IO=0x340 VECTOR=0x49 is typical. Settings are system dependent.

##### ACCTON - EN166X MPX 2 Ethernet

use "1step" program located on the setup disk to set card to jumpered "ne2000" mode.

IO=0x300 VECTOR=0x43 is typical. Settings are system dependent.

##### D-LINK DE-220PCT - 10Mbps Combo 16-Bit Ethernet ISA Adapter

Use "setup" program located on the setup disk "A:\SETUP\setup.exe" to setup the card. Disable PNP and setup Interrupt and I/O base address.

##### Compex - ReadyLink 2000 - PCI 32-bit

Just plug and go. Multiple cards may be used by using the PCI Specific Settings listed below.

#### PCI Specific Settings Information

When using multiple NE2000 PCI cards in a system you may force the driver to use a specific slot or card number for the device being used. PCIINDEX may be used to specify the card instance to be used. Keep in mind the PCIINDEX method is based on a first found basis, so moving cards in the system will change the configuration used. You may also use the PCIBUS and PCIDEV to force the use of the device to a specific slot. To find out the current PCIBUS and PCIDEV values use the OS-9 command *pciv*.

```
/*
 * PCI Specific Settings
 */

#define PCIINDEX0x00/* 0 picks first card */
#define PCIBUS 0x00/* 0 indicates to search */
#define PCIDEV0x00/* 0 indicates to search */
```

## Connection Type

The default connection type is set by either the configuration setup program that came with the card or by hardware jumpers employed. If you are unable to communicate with this card and netstat -in shows the device, the connection type may be incorrect. To correct it, you may do one of the following.

1. Use the NE2000 setup disk to configure the card for the connection used.
2. Change the OS-9 device descriptor for the type of connection in use.
3. Try one of the other connections on the card (if using AUI type, try the RJ45 connector).

## Interrupt Conflict

Another problem may be the interrupt used. The default interrupt is IRQ9. In this case you have the following options.

1. Choose a interrupt that matches the system configuration such as IRQ10 (0x4a). In this case the OS-9 device descriptor must be changed.

If an interrupt conflict exists the device will either not work at all or will hang when the conflicting device is accessed. Mapping the interrupts used in the system is recommended.

Use the command `irqs` to see a list of interrupts.

```
(Super)[/h0/sys/>] irqs
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver   | dev list |
|--------|--------|-------|------------|------------|----------|----------|
| 7      | (\$07) | 10    | \$0003c444 | \$0010f7b4 | fpu      | <na>     |
| 14     | (\$0e) | 1     | \$0003c3a4 | \$00110113 | vectors  | <na>     |
| 64     | (\$40) | 10    | \$00ff40b0 | \$0011098f | tk8253   | <na>     |
| 65     | (\$41) | 10    | \$00ffa680 | \$00120582 | sc8042m  | <na>     |
| 65     | (\$41) | 10    | \$00e85db0 | \$00120582 | sc8042m  | <na>     |
| 65     | (\$41) | 10    | \$00e84a40 | \$00120582 | sc8042m  | <na>     |
| 65     | (\$41) | 10    | \$00e82980 | \$00120582 | sc8042m  | <na>     |
| 74     | (\$49) | 1     | \$00ff02d0 | \$001f9504 | spne2000 | <na>     |
| 78     | (\$4e) | 10    | \$00ff4f30 | \$00137906 | rb1003   | <na>     |

In the case above, we can go into RomBug by typing `break` and placing a breakpoint at the ISR.

```
$ break
RomBug: b 1f9504
RomBug: g
```

and then pinging a machine on the net:

```
$ ping 182.52.109.13
```

( using the actual address of another machine on the network, rather than the one shown above).

If interrupts are running you should be presented a *Rombug* prompt at the breakpoint address. You can type `g` to see if you get another interrupt or `k` to kill the breakpoint.

## Port Address Conflict

It is also possible that the port address used for this card is used by another device in the system. If this is the case, the OS-9 command `netstat -in` will not show the card as available.

The following `netstat` example shows a working network card configured with IP address 182.52.109.25 and MAC address of 00.00.C0.91.4F.96.

```
(Super)[/h0/sys/>] netstat -in
```

| Name  | Mtu  | Network    | Address           | Ipkts | Ierrs | Opkts | Oerrs | Coll |
|-------|------|------------|-------------------|-------|-------|-------|-------|------|
| lo0   | 1536 | <Link>     |                   | 0     | 0     | 0     | 0     | 0    |
| lo0   | 1536 | 127        | 127.0.0.1         | 0     | 0     | 0     | 0     | 0    |
| enet0 | 1500 | <Link>     | 00.00.C0.91.4F.96 | 55    | 110   | 35    | 0     | 0    |
| enet0 | 1500 | 182.52.109 | 182.52.109.25     | 55    | 110   | 35    | 0     | 0    |

## How to modify the OS-9 descriptor

Edit the file

MWOS/OS9000/80386/PORTS/PCAT/SPF/NE2000/DEFS/spf\_desc.h.

Change the fields below as required.

```
/*
 * From spf_desc.h
 */

#define PORTADDR0x00000340/* Base address of hardware */
#define VECTOR0x49/* Port vector */

/*
 * PCI Specific Settings
 */

#define PCIINDEX0x00/* 0 picks first card */
#define PCIBUS0x00/* 0 indicates to search */
#define PCIDEV0x00/* 0 indicates to search */
```

Finally, remake the descriptor: change to the MWOS/OS9000/80386/PORTS/PCAT/SPF/NE2000 directory and type:

```
C:> os9make -f=spfdesc.mak
```

Next time you run the Wizard the new descriptor will be used.

## Low-level system changes

If system state debugging is used, you must change the low level system by modifying the following lines from the file

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des:

```
#define LLNE2000_PORT_ADDRESS0x340
#define LLNE2000_IF_VECTOR0x49
```

as required by the system. For example, for IRQ10, here are the changes required.

```
#define LLNE2000_PORT_ADDRESS0x340
#define LLNE2000_IF_VECTOR0x4a
```

The Wizard will automatically re-make the cnfgdata module.

## Cirrus Logic CS8900

### Overview

The OS9 sp8900 software driver provides support for the Cirrus Logic CS8900a Ethernet Controller. This allows the device to be used as part of an OS9 SoftStax network implementation.

The Cirrus Logic CS8900a provides single chip support for IEEE 802.3 Ethernet. It has a direct ISA bus interface and is therefore commonly found in PC-AT type environments.

The OS9 sp8900 driver takes advantage, where appropriate, of the Plug and Play capability of the cs8900a device. This reduces the time taken to configure the cs8900a for use within an OS9 environment.

### Hardware Configuration

#### Basic Configuration

The CS8900a should be supplied with an MSDOS hosted configuration program. This should be used to pre-configure the device for use. This program assumes the cs8900a has the associated EEPROM as recommended. This EEPROM is used to store the cs8900 configuration parameters. At this time OS9 will only support devices that have this configuration.

#### Using the Setup Program.

Before using the setup program, the user should determine the network adaptor's IO address and Interrupt level. The cs8900 has a limited number of possible combinations, these should be chosen with care. As a default OS9 will assume IO port 0x300 and IRQ Level 10. It is also important to note that OS9 drives the device using the PC-AT I/O Bus for ALL operations. Therefore shared memory should be disabled for OS9 operation.

Having selected the correct choices the user may run the setup program and configure the cs8900 accordingly.

If the device was supplied without a configuration utility it will be necessary to obtain this from the vendor or try the cirrus logic Web site at <http://www.cirrus.com/drivers/>

The setup program also incorporates a self test utility that may be used to confirm correct operation of the device before proceeding.

## OS9 Software Configuration

### Configuring PnP Firmware

The OS9 sp8900 driver will use the PnP (Plug and Play) capability of the cs8900a. This will only be used if it is enabled in the OS9 device descriptor. When enabled the OS9 driver will search all possible I/O locations for a cs8900a device. If found, the first one, starting at the lowest valid I/O address, will be used. The software will confirm that the EEPROM is present. The OS9 driver extracts the necessary configuration details from this device and initializes the cs8900a.

### Configuring OS9 Descriptors

The OS9 device descriptor allows the user to override the PnP default configuration. At this time only a subset of all the possible configuration parameters may be overridden. To change the PnP values the following fields must be modified. This should be performed using a text editor and the OS9 tools provided within the Microware Hawk package.

Once modified the descriptor should be regenerated and tested.

### Device Descriptor Fields

The standard device descriptor is as follows. This file may be found in

```
.../MWOS/OS9000/<processor>/PORTS/<port>/SPF/SP8900/DEFS/spf_desc.h
#define SPF_DIR_NONE0xFF
#define SPF_DIR_IN0x00
#define SPF_DIR_OUT0x01

#include <SPF/item.h>

#ifdef spcs0

/** Device Descriptor for SPF 8900 ethernet driver */
#define PNPON 1 /* do plug and play */
#define PNPOFF 0 /* Use descriptor values (see manual) */

/*****
 * User configuration defines
 *****/
/*****
 * Port configuration defines
 *****/

/* Macros that initialize device descriptor common fields */

/* 300/320/340/360 */

#define PORTADDR0x300/* Base address of hardware */
#define LUN0x7F/* logical unit number */
```

```
#define VECTOR0x4a/* Port vector */
#define PRIORITY8/* IRQ polling priority */
#define IRQLEVEL0/* Port IRQ Level */
#define PNP8900PNPON/* Do plug and play (Normal setting) */

#define TB486COMPATTRUE
-----*/
```

Any information ( not shown ) beyond this point **MUST** not be changed

### User configurable fields

The following fields are user configurable.

| Field Name   | Default Value | Possible Values     |
|--------------|---------------|---------------------|
| PORTADDR     | 0x300         | 0x200..0x360        |
| VECTOR       | 0x4a          | 0x45,0x4a,0x4b,0x4c |
| PRIORITY     | 0x08          | 0..255              |
| PNP8900      | PNPON         | PNPON or PNPOFF     |
| TB486COMPAT* | TRUE          | TRUE or FALSE       |

\*note: The tb486 board is a special case and this flag should be set false for any other board type.

### Generating a New Device Descriptor

Having located and edited the field as desired the new device descriptor may be generated with the following steps

Change directory to:

../MWOS/OS9000/80386/PORTS/PCAT/SPF/SP8900

Enter the command: `os9make -f=sppfdesc.mak -u MOPTS=-u`

The new descriptor will be built.

### OS9 SP8900 Components

The complete driver consists of two OS9 load modules. Users should refer to the appropriate Microware manual for further information concerning system configuration.

The SP8900 component files are :

SP8900 -- cs8900a Ethernet Driver

spcs0 -- cs8900 Device Descriptor



## Sequential Device Support

### VGA Graphics / Keyboard

VGA support is provided using standard VGA graphics screen and keyboard. Most PC based systems use VGA keyboard as the default device for user input. While this is not required for OS-9 based systems it is a convenient way to initially setup systems for use with OS-9.

During the development of MAUI user applications, a serial console may be the preferred method since the text based console may interfere with the graphics application on the same device.

MULTI-TERM is a feature of the VGA Graphics/Keyboard console driver which provides up to four virtual screens. Console users may switch between screens by pressing an alternate function key combination, such as <Alt> <F1>, <Alt> <F2>, <Alt><F3> or <Alt><F4>. MULTI-TERM may be started automatically in the /h0/sys/startup file or manually from the console by executing the following commands:

```
$ mshell -l <>>>/mterm1&
$ mshell -l <>>>/mterm2&
$ mshell -l <>>>/mterm3&
```

### VGA TERMINAL Descriptors Notes

```
/mterm0Multi-term descriptor 0
/mterm1Multi-term descriptor 1
/mterm2Multi-term descriptor 2
/mterm3Multi-term descriptor 3
```

The following optional settings apply to the VGA/Keyboard console:

```
#define DS_ROMBREAK1/* Enter RomBug - Shift PrintScreen. */
#define DS_RESTART1/* Reset System - Ctrl/Alt/Del. 0=disabled */
#define DS_NUM_LOCK1/* Keyboard Number lock 0=off 1=on */
#define DS_SHIFT_LOCK0/* Keyboard Caps lock 0=off 1=on */
```

To change these options, edit the file MWOS/OS9000/80386/PORTS/PCAT/SCF/SC8042M/config.des. Find the sections as outlined above. Change as desired. Then, change to the MWOS/OS9000/80386/PORTS/PCAT/SCF/SC8042M/DRVR directory and type os9make.

## Language support options

To change the language support for the keyboard use the advanced mode from the Wizard and select BOOTFILE OPTIONS tab. Select the language desired.

```
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\term0
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm0
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm1
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm2
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm3
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\term0_fr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm0_fr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm1_fr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm2_fr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm3_fr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\term0_gr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm0_gr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm1_gr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm2_gr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm3_gr
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\term0_nw
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm0_nw
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm1_nw
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm2_nw
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm3_nw
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\term0_uk
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm0_uk
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm1_uk
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm2_uk

MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC8042M\mterm3_uk
```

## Serial Mouse

Configuration modules for a Serial Mouse is included in the system image when the Mouse option is not selected in the Configuration Wizard's Master Builder screen. Serial mouse support is only included when sc16550 support is enabled in the Configuration Wizards BOOTFILE OPTIONS dialog box.

The default port is COM1. The MWOS/OS9000/80386/PORTS/PCAT/BOOTS/INSTALL/PORTBOOT/bootfile.ml file may be changed to allow a different port to be used.

Default (Serial Mouse configured using COM1)

\*

```
* [OPTION4 && !MOUSE] serial mouse
*
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t1
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t2
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t3
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t4
*
```

## Changed to use COM3

```
*
* [OPTION4 && !MOUSE] serial mouse
*
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t1
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t2
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t3
* ././././CMDS/BOOTOBJS/DESC/SC16550/m0_t4
```

## PS2 Mouse

PS2 mouse support is automatically included when the Mouse option is selected from the Configuration Wizard's Master Build screen.

## 16550 Serial

Standard PC type serial ports are supported. By default, four descriptors are available, but you may add more as needed.

Use of COM1 and COM2 are standard on PC based systems. COM3 and COM4 are not. Since COM1 and COM2 use IRQ3 and IRQ4, most systems will not allow COM3 and COM4 to also use IRQ3 and or IRQ4. The main reason for this is that IRQ3 and IRQ4 are normally edge based interrupts, and the 16550 is normally implemented in a edge based configuration. Therefore, anytime COM3 and or COM4 are used, the user must determine the interrupt vector to use for these ports.

To change the vector the user must edit the systype.h file located in the port directory.

MWOS/OS9000/80386/PORTS/PCAT/systype.h

```
#define T1BASE_165500x000003f8/* SC16550 port 1 */
#define T1VECT_165500x44/* IRQ 4 */
#define T1PRI_165505/* Priority */

#define T2BASE_165500x000002f8/* SC16550 port 2 */
#define T2VECT_165500x43/* IRQ 3 */
```

```
#define T2PRI_165505/* Priority */

#define T3BASE_165500x000003e8/* SC16550 port 3 */

#define T3VECT_165500x44/* IRQ 4 */
#define T3PRI_1655010/* Priority */

#define T4BASE_165500x000002e8/* SC16550 port 4 */
#define T4VECT_165500x43/* IRQ 3 */
#define T4PRI_1655010/* Priority */
```

## Making the descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory:

MWOS/OS9000/80386/PORTS/PCAT/SCF/SC16550/DESC

Type os9make; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\term1
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\t1
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\term2
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\t2
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\term3
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\t3
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\term4
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\t4
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\ps
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\m0_t1
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\m0_t2
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\m0_t3
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SC16550\m0_t4
```

## Digiboard

Support for the Digiboard intelligent serial card is included by selecting the Digiboard option in the Configuration Wizard's Bootfile Options dialog box.

To change the vector the user must edit the systype.h file located in the port directory.

MWOS/OS9000/80386/PORTS/PCAT/systype.h

```
#define DIGIPOINT 0xe0 /* port address of DIGI board status reg. */
#define DIGILEVEL 0x45 /* 16450 keyboard controller */
#define DIGIVECTOR DIGILEVEL /* irq vector same as irq level */

#define T10PORT 0x320 /* t10 onboard port address */
```

```
#define T11PORT 0x328 /* t11 onboard port address */
#define T12PORT 0x330 /* t12 onboard port address */
#define T13PORT 0x338 /* t13 onboard port address */
#define T14PORT 0x340 /* t14 onboard port address */
#define T15PORT 0x348 /* t15 onboard port address */
#define T16PORT 0x350 /* t16 onboard port address */
#define T17PORT 0x358 /* t17 onboard port address */
```

## Making the descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory:

MWOS/OS9000/80386/PORTS/PCAT/SCF/SCPC8/DESC

Type *os9make*; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t10
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t11
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t12
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t13
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t14
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t15
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t16
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCPC8\t17
```

## Hostessl

Support for the Hostessl intelligent serial card is included by selecting the Hostessl option in the Configuration Wizard's Bootfile Options dialog box.

To change the vector the user must edit the systype.h file located in the port directory.

MWOS/OS9000/80386/PORTS/PCAT/systype.h

```
#define HS_PORT0x00000218/* Hostess i board. serial adapter board */
#define HS_VECTOR0x4f/* IRQ 15 */
#define HS_BOARDMEM0xd0000/* onboard memory place in the system address space
*/
#define HS_NBLINES16/* number lines on the board (8/16) */

/* Old board doesn't permit 16 bits mode. */
#define HS_BUSSIZE8/* size of the bus the board uses (8/16) */
```

## Making the descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory:

MWOS/OS9000/80386/PORTS/PCAT/SCF/SCHOST/DESC

Type `os9make` the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t40
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t41
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t42
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t43
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t44
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t45
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t46
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t47
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t48
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t49
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t50
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t51
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t52
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t53
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t54
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCHOST\t55
```

## Risicom

Support for the Risicom8 intelligent serial card is included.

To change the vector the user must edit the systype.h file located in the port directory.

MWOS/OS9000/80386/PORTS/PCAT/systype.h

```
#define RC8BASE0x00000220/* Risicom8 serial port adapter */
#define RC8VECT0x45/* IRQ 5 */
```

## Making the descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory:

MWOS/OS9000/80386/PORTS/PCAT/SCF/SCPC8/DESC

Type `os9make`; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCRISCOM\t20
MWOS\OS9000\80386\PORTS\PCAT\CMD\BOOTOBJS\DESC\SCRISCOM\t21
```

```
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t22
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t23
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t24
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t25
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t26
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCRISCOM\t27
```



## WARNING

We have not been able to obtain a Riscom8 board to verify this driver with this release.

## Parallel Printer

Standard PC style printer support is included.

To change the vector or port address the user must edit the systype.h file located in the port directory.

MWOS/OS9000/80386/PORTS/PCAT/systype.h

```
#define PLEVEL0x47/* scp87303 parallel port */
#define PVECTPLEVEL/* irq vector same as irq level */
#define LPT1BASE0x000003bc/* base address of first parallel port */
#define LPT2BASE0x00000378/* base address of second parallel port */
#define LPT3BASE0x00000278/* base address of third parallel port */
```

## Making the descriptors

Once the systype.h file has been updated the new descriptors may be created.

Change to directory:

MWOS/OS9000/80386/PORTS/PCAT/SCF/SCP87303/DESC

Type os9make; the following descriptors will be made:

```
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCP87303\p.lp1
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCP87303\p.lp2
MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\SCP87303\p.lp3
```

## Physical Disk Media

### IDE Standard

Support for IDE based devices, including standard IDE based hard disk. Primary and secondary controllers with master and slave drive support. On some embedded systems Compact Flash supported devices may be used as if they were standard PC AT based devices.

#### Benefits

- Supports large media (8.5GB maximum).
- PIO mode three supported.
- PC File system supported including long filenames (FAT32 is not supported). Boot support (requires OS-9 coreboot load).
- Native RBF file system supported. Full boot support including IPL boot technology.

The standard configuration assumes the primary controller is located at 0x1f0 with IRQ 14 and secondary controller at 0x170 with IRQ 15. The user may, however, change these values as needed to suit the target. The values are based on the contents of the files MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des and MWOS/OS9000/80386/PORTS/PCATsystype.h.

The pertinent lines in MWOS/OS9000/80386/PORTS/PCATsystype.h are:

```
#if defined(RB1003_SPEC_IO_ADDRESS) /* PCMCIA */

#defineBASE_RB1003_PRI 0x00000320/* IDE controller port addr */
#defineVECT_RB1003_PRI0x0/* IDE controller vector */
#defineBASE_RB1003_SEC0x00000360/* IDE 2nd controller port */
#defineVECT_RB1003_SEC0x0/* IDE 2nd controller vector */

#else

#defineBASE_RB1003_PRI0x000001f0/* IDE controller port addr */
#defineVECT_RB1003_PRI0x4e/* IDE controller vector */
#defineBASE_RB1003_SEC0x00000170/* IDE 2nd controller port */

#defineVECT_RB1003_SEC0x4f/* IDE 2nd controller vector */

#endif
```



while in MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des  
the portion of interest is

```
/*
 * Device specific defines
 *
 * ds_idetype = IDE interface type
 * IDE_TYPE_STANDARD
 * IDE_TYPE_PCI
 * IDE_TYPE_PCMCIA
 *
 * ds_polled = IDE_POLLED
 * IDE_INTERRUPTS
 *
 * ds_altstat = HD_DEFAULT_ALTSTAT (Standard IDE offset)
 *
 * HD_PCMCIA_ALTSTAT (PCMCIA IDE offset)
 *
 * ds_timeout = Drive ready timeout in seconds.
 * IDE specification allows for up to
 * 30 seconds. We will allow the max here.
 * Users are free to reduce this amount
 * if desired. PCMCIA IDE FLASH type cards
 * require only a few milliseconds. Rotating
 * devices will require more time.
 */

#define IDE_TYPE_STANDARD 0

#define IDE_TYPE_PCI 1
#define IDE_TYPE_PCMCIA 2

#define IDE_INTERRUPTS 0
#define IDE_POLLED 1

#define HD_DEFAULT_ALTSTAT 0x0206
#define HD_PCMCIA_ALTSTAT 0xe

init dev_specific {

#if defined(RB1003_SPEC_IO_ADDRESS)
 ds_idetype = IDE_TYPE_PCMCIA;
 ds_polled = IDE_POLLED;
 ds_altstat = HD_PCMCIA_ALTSTAT;
 ds_timeout = 30;

#else

 ds_idetype = IDE_TYPE_STANDARD;
 ds_polled = IDE_INTERRUPTS;
 ds_altstat = HD_DEFAULT_ALTSTAT;
```

```
ds_timeout = 30;

#endif
};
```



## Note

Since OS-9 does not require the BIOS to use IDE it is possible on some systems to use IDE without interrupts. Keep in mind that on some systems disabling the IDE from the BIOS also disables the IDE controller as well.

Drive time-out may also fail on drives that are extremely old. If you are having problems using drives that are less than 540MB you may want to disable the time-out. This can be done by setting time-out value to zero in config.des and re-making the descriptors and boot image.

## Using IDE in PCI mode (Advanced - Optional)

Support is included to support IDE devices as PCI specific devices. PCI based IDE support is not automatic and may not work on some PCI bridges. The rb1003 driver must be re-made with the following changes to the makefile.

```
PCILIB = -l=$(PORT)/LIB/pcilib.l

LIB = $(PICLIB) $(PCILIB) \
 $(CPULIB) $(CLIB) $(P2LIB) $(OS_LIB) $(SYS)

SPEC_COPTS = -a -c -r -t=0 -bepg -dNEWINFO $(PICISR) $(IRQMASK) \
 -dPCI
```

In this case we have added PCILIB as well as defined PCI in the SPEC\_COPTS section. On some systems that use both primary and secondary controllers that allow level interrupt to be set and used in PCI standard method, you can save one interrupt vector. You must also set

the device type to PCI in the config.des file shown above. You must have the sources for RB1003 for the ability to make this change using the cross hosted utilities.

If the PCI bridge does not work in PCI mode you can modify the RB1003 init code as need for the PCI bridge device used. The sources are located in MWOS/OS9000/SRC/IO/RBF/DRVR/RB1003, and are included with the Embedded Systems package.

Use of IDE in PCI mode adds about 2K to the driver size.

## RBF

OS-9 RBF native file system may be used on any IDE drive. For more information see BootGen and [IDE Descriptors](#).

## PCF

A PC style file system is also supported. FAT32, however, is not supported in this release. If access to partitions other then the primary are required you may use the pinfo utility to obtain the information required to create specific device descriptors. For more information see [IDE Descriptors](#). You may select the PCF file system as the boot media.

For example. If the drive is Fat (not Fat32) you may place the bootfile image on the root. Make sure it is called os9kboot. Next, create a CMDS and SYS directory at the root level. Copy whatever CMDS you need to the CMDS directory. Create a startup and or password file as needed. This method allows you to use the same partition as Windows95 or NT when you actually run OS-9.

Prepare Windows95/NT based system for use with OS-9.

```
md C:\CMDS

md C:\SYS

copy MWOS\OS9000\80386\CMDS* C:\CMDS

copy MWOS\OS9000\80386\PORTS\PCAT\BOOTS\INSTALL\SYS\MSHELL\startup C:\SYS

copy MWOS\OS9000\80386\PORTS\PCAT\BOOTS\INSTALL\SYS\MSHELL\password C:\SYS

cd C:\SYS

cudo -cdo startup
```

```
cudo -cdo password
```

Although RBF is the preferred file system for use with OS-9 the convenience of using FAT file systems should be taken into consideration when deciding how you want to setup your system.

### Special Note

In the following example the IDE device for /h0 and /dd is set for IDE primary partition four.

If the init dialog is set to /h0 the following is generated. In this case we also have SoftStax SPF enabled.

```
setenv SHELL mshell; alias /dd /hc4;chd /h0 ; chx /h0/cmds;mbinstall;

ipstart;inetd <>>>/nil&;/h0/sys/startup &\n
```

If the init dialog is set to /dd the following is generated. In this case we also have SPF enabled.

```
setenv SHELL mshell; alias /dd /hc4;chd /dd ; chx /dd/cmds;mbinstall ;

ipstart;inetd <>>>/nil&;/dd/sys/startup &\n
```

In both cases the script file on hc4 in sys/startup will be executed. When building systems this file must exist, but does not have to contain data. The following commands suffice to create the expected directory and file:

```
$ mkdir /hc4/SYS
$ touch /hc4/startup
```

It is usually best to create the initial boot image to not use /h0. /dd should be set for RAM disk. This will allow downloading the TAR images. Next setup the final boot image and select /h0 as initial device name.

### Descriptors

Refer to [IDE Descriptors](#) for information on descriptor naming conventions. The descriptors for RB1003 are located in MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBS/DESC/RB1003. Also the RB1003 driver is located in MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBS.

## ROM BOOTING

If changes to the IDE addresses of time-out values are employed, then the ROM boot system may also require changes.

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des

Fixed the following sections:

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=30"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=30"
```

To remove time-out for example we could change the above to:

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=0"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=0"
```

Or we could make the time-out shorter. IDE specification indicates we can wait up to 30 seconds.

```
#define IDE_PRIMARY_PARAMS_PORT"port=0x1f0 timeout=5"
#define IDE_SECONDARY_PARAMS_PORT"port=0x170 timeout=2"
```

## Advanced notes

Some embedded systems include support for Compact Flash, which looks like a standard IDE device. In these cases, we may decide that RBF is the file system of choice, since we can boot the embedded board with no other boot devices installed. How do we place the RBF file system on such small embedded systems? Compact Flash devices will work in PCMCIA systems with a carrier, so that we can use a standard PC with PCMCIA support to build up the PCMCIA disk. Once the disk is built, we can then remove the Compact Flash from the carrier and place it in the target system for use.

## PCMCIA IDE

Support for IDE based devices including standard PCMCIA IDE based hard disk.

### Benefits

- Supports large media(8.5GB maximum).
- PIO mode three supported.
- PC File system supported including long filenames (FAT32 is not supported). Boot support (requires OS-9 coreboot load).

- Native RBF file system supported. Full boot support including IPL boot technology (PCMCIA BIOS BOOT support required if this option is used).
- Requires no interrupts. Interrupts are optional.

The standard configuration assumes socket #0 is mapped to 0x320 and socket #1 is mapped to 0x360. The default configuration does not use interrupts. Users may however enable interrupts if desired.

Example (Enable interrupts on PCMCIA device in socket #0 only - IRQ5 used)

```
/*
 * MWOS/OS9000/80386/PORTS/PCATsystype.h file.
 */

#defineBASE_RB1003_PRI0x00000320/* IDE controller port addr */
#defineVECT_RB1003_PRI0x45/* IDE controller vector */

/*
 * MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des
 */

ds_idetype = IDE_TYPE_PCMCIA;
ds_polled = IDE_INTERRUPTS;
ds_altstat = HD_PCMCIA_ALTSTAT;
ds_timeout = 30;

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des

#define IDE_CIS_PARAMS "ide0=0x320,5 ide1=0x360,0"
```

Once the changes are made change to the MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/DESC directory and type os9make. The changes to MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des are automatically taken care of next time you run the Wizard.

Changes to the default values are based on the MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des file as well as the MWOS/OS9000/80386/PORTS/PCATsystype.h file.

MWOS/OS9000/80386/PORTS/PCATsystype.h

```
#if defined(RB1003_SPEC_IO_ADDRESS) /* PCMCIA */

#defineBASE_RB1003_PRI0x00000320/* IDE controller port addr */
```

```

#define VECT_RB1003_PRI0x0 /* IDE controller vector */
#define BASE_RB1003_SEC0x00000360 /* IDE 2nd controller port */
#define VECT_RB1003_SEC0x0 /* IDE 2nd controller vector */

#else

#define BASE_RB1003_PRI0x000001f0 /* IDE controller port addr */
#define VECT_RB1003_PRI0x4e /* IDE controller vector */

#define BASE_RB1003_SEC0x00000170 /* IDE 2nd controller port */
#define VECT_RB1003_SEC0x4f /* IDE 2nd controller vector */

#endif

MWOS/OS9000/80386/PORTS/PCAT/RBF/RB1003/config.des

/*
 * Device specific defines
 *
 * ds_idetype = IDE interface type
 * IDE_TYPE_STANDARD
 * IDE_TYPE_PCI
 * IDE_TYPE_PCMCIA
 *
 * ds_polled = IDE_POLLED
 * IDE_INTERRUPTS
 *
 * ds_altstat = HD_DEFAULT_ALTSTAT (Standard IDE offset)
 *
 * HD_PCMCIA_ALTSTAT (PCMCIA IDE offset)
 *
 * ds_timeout = Drive ready timeout in seconds.
 * IDE specification allows for up to
 * 30 seconds. We will allow the max here.
 * Users are free to reduce this amount
 * if desired. PCMCIA IDE FLASH type cards
 * require only a few milliseconds. Rotating
 * devices will require more time.
 */

#define IDE_TYPE_STANDARD0
#define IDE_TYPE_PCI1
#define IDE_TYPE_PCMCIA2

#define IDE_INTERRUPTS0
#define IDE_POLLED1

#define HD_DEFAULT_ALTSTAT0x0206
#define HD_PCMCIA_ALTSTAT0xe

init dev_specific {

```

```
#if defined(RB1003_SPEC_IO_ADDRESS)
 ds_idetype = IDE_TYPE_PCMCIA;
 ds_polled = IDE_POLLED;
 ds_altstat = HD_PCMCIA_ALTSTAT;
 ds_timeout = 30;

#else

 ds_idetype = IDE_TYPE_STANDARD;
 ds_polled = IDE_INTERRUPTS;
 ds_altstat = HD_DEFAULT_ALTSTAT;
 ds_timeout = 30;

#endif
};
```

## RBF

OS-9 RBF native file system may be used on any IDE drive including PCMCIA devices. For more information see BootGen and IDE DESCRIPTORS. When using RBF with PCMCIA only OS-9 will be able to access the media. When running FDISK on PCMCIA media, be sure to write down the ID type. You will need this value if you decide to later restore the media for use with DOS/ Windows. fdisk -d=/pchcfmt -s will show the type. If you need to restore the PCMCIA IDE card for use with DOS/Windows you must restore the ID type. If you have PCMCIA support at the DOS level you may be able to use FDISK. If not you can use Linux to change the ID type. We may add this feature to OS-9 fdisk in the future but be warned: once the device is changed to RBF if you do not have the tools then this disk will have to stay RBF.

## PCF

PC style file system is also supported. FAT32 is, however, not supported in this release. For more information see [IDE Descriptors](#).

You may select the PCF file system as the boot media.

For example, if the drive is Fat (not Fat32) you may place the bootfile image on the root. Make sure it is called os9kboot. Next create a CMDS and SYS directory at the root level. Copy whatever CMDS you need to the CMDS directory. Create a startup and or password file as needed. This method allows you to use the same partition as Windows95 or NT when you actually run OS-9.

Prepare Windows95/NT based system for use with OS-9.



```
md C:\CMDS

md C:\SYS

copy MWOS\OS9000\80386\CMDS* C:\CMDS

copy MWOS\OS9000\80386\PORTS\PCAT\BOOTS\INSTALL\SYS\MSHELL\startup C:\SYS

copy MWOS\OS9000\80386\PORTS\PCAT\BOOTS\INSTALL\SYS\MSHELL\password C:\SYS

cd C:\SYS

cudo -cdo startup

cudo -cdo password
```

Although RBF is the preferred file system for use with OS-9 the convenience of using FAT file systems should be taken into consideration when deciding how you want to setup your system.

### Special Note

In the following example the IDE device for /h0 and /dd is set for PCMCIA IDE using socket #0.

If the init dialog is set to /h0 the following is generated. In this case we also have SoftStax SPF enabled.

```
setenv SHELL mshell; alias /dd /pcmhcl;chd /h0 ; chx /h0/cmds;mbinstall ;

ipstart;inetd <>>>/nil&:/h0/sys/startup &\n
```

If the init dialog is set to /dd the following is generated. In this case we also have SoftStax SPF enabled.

```
setenv SHELL shell; alias /dd /pcmhcl;chd /dd ; chx /dd/cmds;mbinstall ;

ipstart;inetd <>>>/nil&:/dd/sys/startup &\n
```

In both cases above the script file on hc4 in sys/startup will be executed. When building systems this file must exist but does not have to contain any data. To create the needed directory and file, the following commands suffice:

```
$ mkdir /pcmhcl/SYS

$ touch /pcmhcl/startup
```

It is usually best to create the initial boot image to not use /h0. /dd should be set for RAM disk. This will allow downloading the TAR images. Next setup the final boot image and select /h0 if as initial device name.

## Descriptors

Refer to [IDE Descriptors](#) for information on descriptor naming conventions. The descriptors for RB1003 are located in MWOS/OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RB1003. Also the RB1003 driver is located in MWOS/OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS.

## ROM BOOTING

If changes to the IDE addresses of time-out values are employed then the ROM boot system may also require changes.

MWOS/OS9000/80386/PORTS/PCAT/ROM/cnfgdata.des

Find the following sections:

```
#define IDE_CIS_PARAMS "ide0=0x320,0 ide1=0x360,0"

#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=30 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=30 altstat=0xe"
```

To remove time-out for example we could change the above to:

```
#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=0 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=0 altstat=0xe"
```

Or we could make the time-out shorter. IDE specification indicates we should wait up to 30 seconds.

```
#define PCMCIA_IDE_PRIMARY_PARAMS_PORT"port=0x320 timeout=5 altstat=0xe"
#define PCMCIA_IDE_SECONDARY_PARAMS_PORT"port=0x360 timeout=2 altstat=0xe"
```

To explain the definition of IDE\_CIS\_PARAMS in detail: "ide0=0x320,5 ide1=0x360,0" indicates that IDE0 (socket 0) has a base address of 0x320 and uses IRQ 5, while IDE1 (socket 1) has a base address of 0x360 and uses no interrupt.

## Advanced notes

Some embedded systems support Compact Flash, which looks like a standard IDE device. In these cases, we may decide that RBF is the file system of choice, since we can boot the embedded board with no other

boot devices installed. How do we place the RBF file system on such small embedded systems? Compact Flash devices will work in PCMCIA systems with a carrier, so that we can use a standard PC with PCMCIA support to build up the PCMCIA disk. Once the disk is built, we can then remove the Compact Flash from the carrier and place it in the target system for use.

## IDE Descriptors

For Standard IDE devices the devices are referenced as shown in the following table.

### Standard IDE - RBF Descriptors

|         |                                             |
|---------|---------------------------------------------|
| /hcfmt  | IDE primary master - Entire disk            |
| /hc1fmt | IDE primary master - Primary partition #1   |
| /hc2fmt | IDE primary master - Primary partition #2   |
| /hc3fmt | IDE primary master - Primary partition #3   |
| /hc4fmt | IDE primary master - Primary partition #4   |
| /hdfmt  | IDE primary slave - Entire disk             |
| /hd1fmt | IDE primary slave - Primary partition #1    |
| /hd2fmt | IDE primary slave - Primary partition #2    |
| /hd3fmt | IDE primary slave - Primary partition #3    |
| /hd4fmt | IDE primary slave - Primary partition #4    |
| /hefmt  | IDE secondary master - Entire disk          |
| /he1fmt | IDE secondary master - Primary partition #1 |
| /he2fmt | IDE secondary master - Primary partition #2 |
| /he3fmt | IDE secondary master - Primary partition #3 |
| /he4fmt | IDE secondary master - Primary partition #4 |
| /hffmt  | IDE secondary slave - Entire disk           |
| /hf1fmt | IDE secondary slave - Primary partition #1  |

|         |                                            |
|---------|--------------------------------------------|
| /hf2fmt | IDE secondary slave - Primary partition #2 |
| /hf3fmt | IDE secondary slave - Primary partition #3 |
| /hf4fmt | IDE secondary slave - Primary partition #4 |

### **Standard IDE - PCF Descriptors**

|        |                                           |
|--------|-------------------------------------------|
| /mhc1  | IDE primary master - Primary partition #1 |
| /mhc2f | IDE primary master - Primary partition #2 |
| /mhc3  | IDE primary master - Primary partition #3 |
| /mhc4  | IDE primary master - Primary partition #4 |

|       |                                          |
|-------|------------------------------------------|
| /mhd1 | IDE primary slave - Primary partition #1 |
| /mhd2 | IDE primary slave - Primary partition #2 |
| /mhd3 | IDE primary slave - Primary partition #3 |
| /mhd4 | IDE primary slave - Primary partition #4 |

|       |                                             |
|-------|---------------------------------------------|
| /mhe1 | IDE secondary master - Primary partition #1 |
| /mhe2 | IDE secondary master - Primary partition #2 |
| /mhe3 | IDE secondary master - Primary partition #3 |
| /mhe4 | IDE secondary master - Primary partition #4 |

|       |                                            |
|-------|--------------------------------------------|
| /mhf1 | IDE secondary slave - Primary partition #1 |
| /mhf2 | IDE secondary slave - Primary partition #2 |
| /mhf3 | IDE secondary slave - Primary partition #3 |
| /mhf4 | IDE secondary slave - Primary partition #4 |

### **CDROM IDE Descriptors**

|      |                      |
|------|----------------------|
| /cd0 | IDE secondary master |
|------|----------------------|

### **PCMCIA IDE - RBF Descriptors**

|          |                                    |
|----------|------------------------------------|
| /pchcfmt | PCMCIA IDE Socket #0 - Entire disk |
|----------|------------------------------------|

/pchc1fmt    PCMCIA IDE Socket #0 - Primary partition #1  
/pchefmt    PCMCIA IDE Socket #1 - Entire disk  
/pche1fmt    PCMCIA IDE Socket #1 - Primary partition #1

### **PCMCIA IDE - PCF Descriptors**

/pcmh1c    PCMCIA IDE Socket #0 - Primary partition #1  
/pcmh1e    PCMCIA IDE Socket #1 - Primary partition #1



### **Note**

The descriptors for IDE are automatically included when using the Wizard. Users may also access the descriptors in the MWOS directory structure at:

`MWOS/OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RB1003`

## **DiskOnChip**

M-Systems' DiskOnChip™ is a new generation of single-chip flash disks. The DiskOnChip device contains built-in firmware which provides full hard disk emulation and allows the DiskOnChip to operate as a boot device.

When used under OS-9, the DiskOnChip is managed by a TrueFFS™, technology based device driver, attached to the standard OS-9 file system (RBF) or to a DOS compatible file system (PCF).

This section is intended for systems integrators designing with the DiskOnChip 2000, DiskOnChip Millennium or DiskOnChip DIMM and describes how to use the DiskOnChip as a bootable data storage device under the OS-9 Operating System. In the remainder of this application note the term DiskOnChip is used to describe the above mentioned DiskOnChip Family Products.

## Benefits

- Small footprint for embedded boards.
- Native RBF file system supported. Full boot support including IPL boot technology.

## Low-Level Boot Support

OS9000/80386/CMD5/BOOTOBJS/ROM/doc

## High-Level Support

OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/rbdoc

## Descriptors used by Wizard:

```
OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RBDOC/dochcfmt
OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RBDOC/dochc1
OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RBDOC/dochc1fmt
OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RBDOC/dochc1.h0
```

Additional descriptors are provided in the "OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RBDOC" directory. Please refer to [DiskOnChip Descriptors](#) for information on the use of these other descriptors.

## Building a DiskOnChip Boot Image

The DiskOnChip distribution is made up of two primary modules: *doc*, which is the OS-9 low-level booter module, and *rbdoc*, the OS-9 device driver for DiskOnChip. The following procedure uses the Configuration Wizard to configure a boot device using DiskOnChip. The DiskOnChip device appears as a disk drive to the high-level system and is accessed using the RBF descriptor /dochc1.

1. On your host PC, select Microware Configuration Wizard from the Program Menu.
2. At the main screen of the wizard, select the radio button labeled Advanced Mode.
3. In the text box labeled Configuration Name (Required), type DiskOnChip, Click on the OK button. A new window appears with a menu bar. This is the Advanced Mode window of the Configuration Wizard. It is here that we must tell the wizard to include DiskOnChip support.

4. Navigate to Configure->Coreboot->Disk Configuration. A window appears with several tabs.
5. Select the tab labeled *IDE Configuration*.
6. Look for the area of the window labeled DiskOnChip and click on the checkmark boxes labeled *Add to menu* and *Auto Boot*. This will cause the booter to include the proper modules for allowing the system to boot from the DiskOnChip device.
7. Click on the OK button to dismiss the window.
8. Navigate to Configure->Bootfile->Disk Configuration. A window appears with several tabs.
9. Select the tab labeled *IDE Configuration*.
10. Look for the area of the window labeled DiskOnChip and click on the checkmark box labeled *Enable DOC*. This will cause the booter to include the proper modules for allowing us to format the DiskOnChip device.
11. Click on the tab labeled Init Options.
12. Click on the OK button to dismiss the window.
13. Navigate to Configure->Build Image and another window appears.
14. Click on the *Check* button, then the *Build* button to build the coreboot and bootfile.
15. Once this process is finished, click on the *MakeBoot* button and follow the prompts to create a bootable OS-9 floppy.

### Booting OS-9 and Formatting DiskOnChip

Once the boot floppy has been created, insert it into the target's floppy drive and boot OS-9. Once the shell prompt (\$) appears, format the DiskOnChip device:

```
$ chd /d0
$ iniz /dochcfmt
$ format /dochcfmt -nv -np -r -v
```

### Creating the OS-9 Partition

Start fdisk by typing:

```
$ fdisk -d=/dochcfmt
```

From fdisk, select option 6 to create the Master Boot Record. Respond with y <ENTER>, then <ESC> to return to the fdisk main menu

Use option 1 to create an OS-9000 partition. Press <ENTER> to accept the partition size, then selection option 1 to create an OS-9000/386 partition. Press <ESC> to return to the fdisk main menu.

At the main menu, select option 2 to make the newly created partition active. You must type the partition number (1 in this case) and press <ENTER>. Press <ESC> to return to the fdisk main menu.

Press <ESC> to exit fdisk. You will be prompted for confirmation to change the disk. Type y <ENTER>

### Formatting the Partition

Now the DiskOnChip contains an OS-9000 partition. Format this partition with the following command:

```
$ format /dochclfmt -np -nv -r -v
```

### Installing Boot Files on DiskOnChip

Use the following commands on your OS-9 computer to install the boot image onto the DiskOnChip device.

```
$ bootgen -iipldnoq -lfirstboot /dochclfmt
$ bootgen /dochclfmt sysboot
```

### Booting OS-9 from the DiskOnChip

Booting OS-9 from the DiskOnChip device allows you to use the DiskOnChip as the only disk in the system, holding the operating system itself in addition to all other applications and files.

Please follow the steps described in the following paragraphs in order to use the DiskOnChip as the boot device.

### Updating the DiskOnChip Firmware

By default, the DiskOnChip firmware installs the DiskOnChip as an additional disk in the system. This default allows you to boot an Operating System from the DiskOnChip on a diskless machine. In case your machine is equipped with other hard disk(s) and you still want to boot from the DiskOnChip, you need to install the DiskOnChip as the first drive.

In order to install the DiskOnChip as the first drive, boot your target system into MS-DOS and perform the following command:



```
DUPDATE /WIN:{address} /S:DOC121.EXB /FIRST
```

With the {address} being the base address of the DiskOnChip, i.e. D000, D400, etc. “121” in the file DOC121.EXB represents the firmware version. The actual firmware version used might be a higher version, i.e. DOC122.EXB, etc.



## Note

The DUPDATE utility and firmware files are provided with the DiskOnChip ISA evaluation board available from M-Systems.

The default base address for the M-System's evaluation board is D000h. Refer to the documentation included with your hardware for the base address and board jumper settings.

If you do not need to access additional hard disk(s) under OS-9, you may also disable them in the CMOS setup. In this case, the above DOS command is not necessary.

In some cases it is useful to prevent the DiskOnChip firmware from installing at boot time. You can achieve this by performing the following DOS command:

```
DUPDATE /WIN:{address} /S:DOC2.FFF
```

With the {address} being the base address of the DiskOnChip, i.e. D000h, D400h, etc.

## Booting OS-9 from DiskOnChip

Remove any floppy disks from the floppy drive and reset your target. The OS-9 IPL message should appear briefly, then the following screen:

```
OS-9000/x86 Bootstrap
Now trying to Override autobooters.
```

At this point, the floppy booter will be called and will fail because there is no floppy disk in the drive. At that point, the DiskOnChip booter will read the OS-9 bootfile from the DiskOnChip device. The system will then boot to a shell prompt.

Your system is now fully booting from DiskOnChip!

## DiskOnChip Descriptors

In the MWOS\OS9000\80386\PORTS\PCAT\CMD5\BOOTOBJS\DESC\RBD OC directory there are numerous device descriptors for both RBF and PCF filesystems. Note that the table below omits descriptors with the filename extension .h0 - these files are also present, and contain device descriptors with the canonical name h0, useful for systems whose main disk unit will be a DiskOnChip device.

### DiskOnChip - RBF Descriptors

|            |                                            |
|------------|--------------------------------------------|
| /dochcfmt  | DiskOnChip Device 0 - Entire disk          |
| /dochc1fmt | DiskOnChip Device 0 - Primary partition #1 |
| /dochc2fmt | DiskOnChip Device 0 - Primary partition #2 |
| /dochc3fmt | DiskOnChip Device 0 - Primary partition #3 |
| /dochc4fmt | DiskOnChip Device 0 - Primary partition #4 |
| /dochdfmt  | DiskOnChip Device 1 - Entire disk          |
| /dochd1fmt | DiskOnChip Device 1 - Primary partition #1 |
| /dochd2fmt | DiskOnChip Device 1 - Primary partition #2 |
| /dochd3fmt | DiskOnChip Device 1 - Primary partition #3 |
| /dochd4fmt | DiskOnChip Device 1 - Primary partition #4 |
| /dochefmt  | DiskOnChip Device 2 - Entire disk          |
| /doche1fmt | DiskOnChip Device 2 - Primary partition #1 |
| /doche2fmt | DiskOnChip Device 2 - Primary partition #2 |
| /doche3fmt | DiskOnChip Device 2 - Primary partition #3 |
| /doche4fmt | DiskOnChip Device 2 - Primary partition #4 |
| /dochffmt  | DiskOnChip Device 3 - Entire disk          |
| /dochf1fmt | DiskOnChip Device 3 - Primary partition #1 |
| /dochf2fmt | DiskOnChip Device 3 - Primary partition #2 |

/dochf3fmt DiskOnChip Device 3 - Primary partition #3

/dochf4fmt DiskOnChip Device 3 - Primary partition #4

### **DiskOnChip - PCF Descriptors**

/docmhcfmt DiskOnChip Device 0 - Entire disk

/docmhc1fmt DiskOnChip Device 0 - Primary partition #1

/docmhc2fmt DiskOnChip Device 0 - Primary partition #2

/docmhc3fmt DiskOnChip Device 0 - Primary partition #3

/docmhc4fmt DiskOnChip Device 0 - Primary partition #4

/docmhdfmt DiskOnChip Device 1 - Entire disk

/docmhd1fmt DiskOnChip Device 1 - Primary partition #1

/docmhd2fmt DiskOnChip Device 1 - Primary partition #2

/docmhd3fmt DiskOnChip Device 1 - Primary partition #3

/docmhd4fmt DiskOnChip Device 1 - Primary partition #4

/docmhcfmt DiskOnChip Device 2 - Entire disk

/docmhe1fmt DiskOnChip Device 2 - Primary partition #1

/docmhe2fmt DiskOnChip Device 2 - Primary partition #2

/docmhe3fmt DiskOnChip Device 2 - Primary partition #3

/docmhe4fmt DiskOnChip Device 2 - Primary partition #4

/docmhffmt DiskOnChip Device 3 - Entire disk

/docmhf1fmt DiskOnChip Device 3 - Primary partition #1

/docmhf2fmt DiskOnChip Device 3 - Primary partition #2

/docmhf3fmt DiskOnChip Device 3 - Primary partition #3

/docmhf4fmt DiskOnChip Device 3 - Primary partition #4

### **PC AT Style Floppy**

Standard floppy support is provided using the RB765 driver. /d0 may be used to access RBF native file system. /md0 may be used to access PC style floppy devices.

See the section **Using Cross Hosted Utilities** for creating and moving data to RBF floppies from Windows95/98 or NT4.0.

## Floppy Descriptors

### Floppy - RBF Descriptors

/d0 Floppy drive A:

### Floppy - PCF Descriptors

/md0 Floppy drive A:



---

### Note

When using the Wizard the descriptors for floppy devices are automatically included. Users may also access the descriptors in the MWOS directory structure.

MWOS/OS9000/80386/PORTS/PCAT/CMD5/BOOTOBJS/DESC/RB765

---

## Symbios 810,810A,825,825A and 875 PCI SCSI controllers—Wide, Ultra and Ultra Wide

### Highlights

- Wide support
- Ultra FAST20 support
- SCRIPTS RAM support ( able to run scripts from on-chip RAM )
- Large FIFO enabled
- Increased burst rates to 128 where supported
- Special PCI cache features enabled
- PCI IO Mode selectable (PCI I/O or PCI Memory )



## Note

The SCRIPTS RAM support is currently only available on OS9000 X86 based systems. Requires non translation of PCI memory. To use SCRIPTS RAM support include the "-dSCRIPTS\_RAM" in the compile line when making the driver.

Instruction prefetch is not enabled by default. Maximum burst rate and large fifo's are enabled.

By default the Microware Symbios driver will use the PCI I/O model. To speed up transfers, especially on X86, platforms the memory module may be used. In the PCI memory mode no in/out instructions are used. For the X86 platform this removes the CPU related waits added by the use of "inc", "outc" etc. If the user desires to run the driver in PCI Memory mode the driver may be recompiled with the "-dPCI\_IO\_MAPPED" flag removed from the

MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

---

```
IO_MAPPED = -dPCI_IO_MAPPED
```

To use memory model change to:

```
IO_MAPPED = # -dPCI_IO_MAPPED
```



## Note

We have changed to default to IO\_MAPPED for X86 due to problems on PCAT based motherboards.

---

Prior to this release the following Symbios devices were supported:

Number of devices supported (2)

```
DEVICEWIDEULTRA1ULTRA2FIFO_SIZEBURST
```

---

Symbios 53c810N/AN/AN/A6416

Symbios 53c825NoN/AN/A8816

This release adds the following:

Number of devices supported (12)

DEVICEWIDEULTRA1ULTRA2FIFO\_SIZEBURST

---

Symbios 53c810N/AN/AN/A6416

Symbios 53c810APN/AN/AN/A6416 (1)

Symbios 53c815N/AN/AN/A6416 (1)

Symbios 53c820YesN/AN/A8816 (1)

Symbios 53c825YesN/AN/A8816

Symbios 53c825AYesN/AN/A536128

Symbios 53c875YesYESN/A536128

Diamond FirePort20YesN/AN/A536128 (825A)

Diamond FirePort40YesYESN/A536128 (875)

Symbios 53c860YesYESN/A536128 (1)

Symbios 53c885YesYESN/A536128 (1)

Symbios 53c895YesYESYES536128 (1,2)

Symbios 53c896YesYESYES536128 (1,2)

(1) Support is included but untested.

(2) Support for 895 and 896 is only available with out ULTRA support.  
The 160Mhz clock will be enabled on a future release.



## Note

The 895 and 896 have not been tested.

---

[Symbios 53C810]

[Symbios 53C810A]

Device supports burst op code fetch

Device supports instruction prefetch

Device supports Cache Line Size and Cache Commands

[Symbios 53C810ALV] \* same as 810

Device supports burst op code fetch

Device supports instruction prefetch  
Device supports Cache Line Size and Cache Commands  
[Symbios 53C815]  
Device supports burst op code fetch  
[Symbios 53C825]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
[Symbios 53C825A]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
[Symbios 53C860]  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers  
[Symbios 53C875]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers  
[Symbios 53C885]  
Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers

Device supports Clock Doubler

[Symbios 53C895]

Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers ( Not supported yet )

Device supports Clock Doubler ( Not supported yet )

Device supports Fast-40 transfers ( Not supported yet )

[Symbios 53C896]

Device supports Wide SCSI data transfers

Device supports burst op code fetch

Device supports instruction prefetch

Device has Scripts RAM

Device supports Cache Line Size and Cache Commands

Device supports Fast-20 transfers ( Not supported yet )

Device supports Clock Doubler ( Not supported yet )

Device supports Fast-40 transfers ( Not supported yet )

Using Ultra Fast20 and Wide support.

### **Controller Dependency:**

For FAST20 support the controller must support FAST20.



**Device Descriptors:**

To use a device with disconnect, wide, synchronous data transfer, and FAST20 Ultra the following should be added to the device descriptor entry in "systype.h". Be sure to re-make the descriptors.

```
#define SCSTOPTS SCSI_ATN|SCSI_SYNC|SCSI_WIDE|SCSI_ULTRA
```

Optionally you may use EditMod to change the SCSTOPTS field. For SYNC and ATN the SCSTOPTS value is "5".

**Using multiple SCSI controllers**

It is possible to use multiple SCSI controllers with the Symbios family of controllers.

The port address is used to specify the card to use.

PortAddress format.

[0xff] [device] [index] [SCSI\_ID]

device = device number. Use PCIV to discover index to match. This is system dependent and slot dependent.

Index = you may instead use index to specify the index of the card found. Zero indicates first card, one indicates second card, etc.

The same address information may be used from the OS-9 boot menu to access additional SCSI controllers, e.g.:

: hs port=0xff000100 id=3 ? Boot from second SCSI controller SCSI ID=3

**Creating driver specific versions**

By default, the Symbios scsi8xx driver will look for any Symbios SCSI card based on table usage. You may however re-compile the driver to only look for the card desired.

MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

PCI\_DEV\_ID = # -dSYMBIOS\_DEVICE\_ID=0xf

Remove the # and specify the ID required.

Driver name: scsi8xx

Rom driver name: ncr8xx

## Diamond FirePort20 and FirePort40—Wide, Ultra and Ultra Wide

### Highlights

- Wide support
- Ultra FAST20 support
- SCRIPTS RAM support ( able to run scripts from on chip RAM ) (1)
- Large FIFO enabled
- Increased burst rates to 128 where supported
- Special PCI cache features enabled (2)
- PCI IO Mode selectable (PCI I/O or PCI Memory ) (3)

### Notes

1. The SCRIPTS RAM support is currently only available on OS-9, X86 based systems. Requires non translation of PCI memory. To use SCRIPTS RAM support include the "-dSCRIPTS\_RAM" in the compile line when making the driver.
2. Instruction prefetch is not enabled by default. Maximum burst rate and large fifo's are enabled.
3. By default the Microware Symbios driver will use the PCI I/O model. To speed up transfers especially on X86 platforms the memory module may be used. In the PCI memory mode no in/out instructions are used. For the X86 platform this removes the CPU related waits added by the use of "inc", "outc" etc... If the user desires to run the driver in PCI Memory mode the driver may be recompiled with the "-dPCI\_IO\_MAPPED" flag removed.

MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

IO\_MAPPED = -dPCI\_IO\_MAPPED

To use memory model change to:

IO\_MAPPED = # -dPCI\_IO\_MAPPED



## Note

We have changed the default to IO\_MAPPED for X86 due to problems on PCAT based motherboards.

Prior to this release the following Symbios devices were supported:

### Number of devices supported (2)

DEVICEWIDEULTRA1ULTRA2FIFO\_SIZEBURST

Symbios 53c810N/AN/AN/A6416

Symbios 53c825NoN/AN/A8816

This release adds the following:

### Number of devices supported (12)

DEVICEWIDEULTRA1ULTRA2FIFO\_SIZEBURST

Symbios 53c810N/AN/AN/A6416

Symbios 53c810APN/AN/AN/A6416 (1)

Symbios 53c815N/AN/AN/A6416 (1)

Symbios 53c820YesN/AN/A8816 (1)

Symbios 53c825YesN/AN/A8816

Symbios 53c825AYesN/AN/A536128

Symbios 53c875YesYESN/A536128

Diamond FirePort20YesN/AN/A536128 (825A)

Diamond FirePort40YesYESN/A536128 (875)

Symbios 53c860YesYESN/A536128 (1)

Symbios 53c885YesYESN/A536128 (1)

Symbios 53c895YesYESYES536128 (1,2)

Symbios 53c896YesYESYES536128 (1,2)

1. Support is included but untested.
2. Support for 895 and 896 is only available with out ULTRA support.  
The 160Mhz clock will be enabled on a future release. Note the 895 and 896 have not been tested.

[Symbios 53C810]

[Symbios 53C810A]

Device supports burst op code fetch

Device supports instruction prefetch  
Device supports Cache Line Size and Cache Commands  
[Symbios 53C810ALV] \* same as 810  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device supports Cache Line Size and Cache Commands  
[Symbios 53C815]  
Device supports burst op code fetch  
[Symbios 53C825]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
[Symbios 53C825A]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
[Symbios 53C860]  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers  
[Symbios 53C875]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM

Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers  
[Symbios 53C885]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers  
Device supports Clock Doubler  
[Symbios 53C895]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers ( Not supported yet )  
Device supports Clock Doubler ( Not supported yet )  
Device supports Fast-40 transfers ( Not supported yet )  
[Symbios 53C896]  
Device supports Wide SCSI data transfers  
Device supports burst op code fetch  
Device supports instruction prefetch  
Device has Scripts RAM  
Device supports Cache Line Size and Cache Commands  
Device supports Fast-20 transfers ( Not supported yet )  
Device supports Clock Doubler ( Not supported yet )

Device supports Fast-40 transfers ( Not supported yet )

Using Ultra Fast20 and Wide support.

### **Controller Dependency:**

For FAST20 support the controller must support FAST20.

### **Device Descriptors:**

To use a device with disconnect, wide, synchronous data transfer, and FAST20 Ultra the following should be added to the device descriptor entry in "systype.h". Be sure to re-make the descriptors.

```
#define SCSTIOPTS SCSI_ATN|SCSI_SYNC|SCSI_WIDE|SCSI_ULTRA
```

Optionally you may use EditMod to change the SCSTIOPTS field. For SYNC and ATN the SCSTIOPTS value is "5".

### **Using multiple SCSI controllers**

It is possible to use multiple SCSI controllers with the Symbios family of controllers.

The port address is used to specify the card to use.

PortAddress format.

[0xff] [device] [index] [SCSI\_ID]

device = device number. Use PCIV to discover index to match. This is system dependent and slot dependent.

Index = you may instead use index to specify the index of the card found. Zero indicates first card, one indicates second card, et. Cetera.

The same address information may be used from the OS-9 boot menu to access additional SCSI controllers, e.g.:

```
: hs port=0xff000100 id=3 ? Boot from second SCSI controller SCSI ID=3
```

### **Creating driver specific versions**

By default, the Symbios scsi8xx driver will look for any Symbios SCSI card based on table usage. You may however re-compile the driver to only look for the card desired.

MWOS/OS9000/80386/PORTS/PCAT/SCSI/SCSI8XX/makefile

```
PCI_DEV_ID = # -dSYMBIOS_DEVICE_ID=0xf
```

Remove the # and specify the ID required.

Driver name: scsi8xx

Rom driver name: ncr8xx

## **Adaptec 1540/1542 ISA**

Support for Adaptec 1540 series is provided, this includes 1540, 1542 and 1542CP. The driver probes the DMA channel used, but the port address and interrupt are fixed. If the vector does not match the card, a Bad Mode error is returned. You may set up the descriptor to use vector zero, which forces the driver to use what the card reports.

```
#defineBASE_AHA15400x00000330
#defineVECT_AHA15400x4b
```

Driver name: aha1540

Rom driver name: ll1540

## **Adaptec 2940, 2940U and 2940UW**

Support for Adaptec PCI series AHA2940, 2940U and 2940UW is provided. Only one SCSI controller of this type is allowed.

Driver name: aic7870

Rom driver name: ll7870

## **SCSI Descriptors**

### **SCSI HARD - RBF Descriptors**

/hs<id>fmt id= SCSI ID (1-f) - Entire disk

/hs<id><part>fmtid= SCSI ID (1-f) part= partition

/hs<id><part> id= SCSI ID (1-f) part= partition

### **SCSI HARD - PCF Descriptors**

/mhs<id><part> id= SCSI ID (1-f) part= partition

### **SCSI FLOPPY - RBF Descriptors**

d<id>\_3.d0      id= SCSI ID (1-f) mapped as drive d0

### SCSI FLOPPY - PCF Descriptors

md<id>\_3.d0      id= SCSI ID (1-f) mapped as drive md0

### SCSI TAPE Descriptors

/mt<id>              id= SCSI ID (1-f)

### SCSI CDROM Descriptors

/cd0                  SCSI ID is set to 5



## Note

When using the Wizard, the descriptors for SCSI are automatically included or created as needed for the SCSI controller selected. Users may also access the descriptors in the MWOS directory structure.

```
MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBS/DESC/SCSI8XX
MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBS/DESC/AHA1540
MWOS/OS9000/80386/PORTS/PCAT/CMDS/BOOTOBS/DESC/AIC7870
```

## System Devices

### Real Time Clock

Real-time clock (RTC) devices with battery backup enable the system clock to be set without operator intervention. The bootfile options dialog in the Configuration Wizard may be used to include one of two possible real-time clock drivers.

The local time driver assumes that the time stored in the RTC device is local time. This option maintains compatibility when another O.S. is installed on the same machine.

The GMT driver assumes that the time stored in the RTC is Greenwich Mean Time.



The driver communicates with the OS-9 kernel using GMT, with the System Time Zone field in the init. module converting between GMT and local time. Refer to the Configuration Wizard Init dialog for information on setting the system time zone.

## Additional Devices

### PPP and SLIP



#### Note

PPP and SLIP is not directly supported by the Wizard.

Although PPP and SLIP are not directly supported by the Wizard, you can use the Wizard to configure and use both PPP and SLIP.

Users may select any or all of Ethernet, PPP, or SLIP. When using PPP or SLIP, the SPF options must be enabled. In the SPF/OPTIONS tab select either SLIP or PPP or both. Make sure SPF is checked when building the boot image. If Ethernet is not desired, select None for the Ethernet controller name.

PPP (How to set up):

Edit the pcat.ini file in the

`MWOS\OS900\80386\PORTS\PCAT\BOOTS\INSTALL\INI`

directory and search for ETHER\_OPTION\_1.

By default, the PPP setup will obtain the address from the server. If desired this may be changed.

`ETHER_OPTION_1=ppp0 binding /ipcp0 iff_pointopoint`

Make sure PPP is selected in the SPF/OPTIONS tab.

Go into the Wizard and select "Enable softstax" just as you would for an Ethernet based connection. Build the boot.

SLIP (How to set up):

Edit the pcat.ini in the directory

```
MWOS\OS900\80386\PORTS\PCAT\BOOTS\INSTALL\INI
```

directory and search for ETHER\_OPTION\_0.

Setup SLIP as required.

```
ETHER_OPTION_0=slip0 address 10.0.0.1 destaddr 10.0.0.2 binding
/spsl0
```

Make sure SLIP is selected in the SPF/OPTIONS tab.

Go into the Wizard and select Enable Softstax just as you would for an Ethernet based connection. Build the boot.

## X86 Utilities

### ABORT

The Abort utility is a p2module that may be used to allow the system to enter debug state once a Non-maskable Interrupt (NMI) is generated.

Usage:

```
$ p2init abort
```

### CACHECHK

The Cachechk utility may be used to verify L2 cache is working on a given system.

```
(Super) [/h0/>] cachechk
```

| Memory Block | Transfer Speed | Access Time   | Chart |
|--------------|----------------|---------------|-------|
| -----        | -----          | -----         | ----- |
| 256          | 292.90 MB/s    | 3.41 ns/byte  | ###   |
| 512          | 301.04 MB/s    | 3.32 ns/byte  | ###   |
| 1024         | 305.30 MB/s    | 3.28 ns/byte  | ###   |
| 2048         | 307.10 MB/s    | 3.26 ns/byte  | ###   |
| 4096         | 307.63 MB/s    | 3.25 ns/byte  | ###   |
| 8192         | 307.10 MB/s    | 3.26 ns/byte  | ###   |
| 16384        | 301.30 MB/s    | 3.32 ns/byte  | ###   |
| 32768        | 176.19 MB/s    | 5.68 ns/byte  | ##### |
| 65536        | 174.72 MB/s    | 5.72 ns/byte  | ##### |
| 131072       | 173.43 MB/s    | 5.77 ns/byte  | ##### |
| 262144       | 164.04 MB/s    | 6.10 ns/byte  | ##### |
| 524288       | 153.46 MB/s    | 6.52 ns/byte  | ##### |
| 1048576      | 96.58 MB/s     | 10.35 ns/byte | ##### |
| 2097152      | 84.34 MB/s     | 11.86 ns/byte | ##### |

In this case we have an L1 cache size of 16K and an L2 cache size of 512K.

### DMPPCI

The DMPPCI utility may be used to examine PCI configuration space.

Usage:

```
(Super) [/h0/>] dmppci -?
```

```
Syntax: dmppci <bus_number> <device_number> <function_number> {<size>}
Function: dump PCI configuration space.
Options:
 none.
(Super)[/h0/>] dmppci 0 4 0
```

```
 PCI DUMP Bus:0 Dev:4 Func:0 Size:64

VID DID CMD STAT CLASS RV CS IL IP LT HT BI MG ML SVID SDID

1013 00d6 0007 00a0 030000 03 00 0a 01 40 00 00 10 10 1013 8000
BASE[0] BASE[1] BASE[2] BASE[3] BASE[4] BASE[5] CIS_P EXROM

e0000000 e2100000 00000000 00000000 00000000 00000000 00000000 00000000
Offset 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

0000 13 10 d6 00 07 00 a0 00 03 00 00 03 00 40 00 00
0010 00 00 00 e0 00 00 10 e2 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 13 10 00 80
0030 00 00 00 00 00 00 00 00 00 00 00 00 0a 01 10 10
```

## GIMMEIO

GIMMEIO is an example trap handler and test program that demonstrates how to allow I/O port access in user state programs. The

MWOS/OS9000/80386/PORTS/PCAT/UTILS/GIMMEIO

directory contains both the test program and trap handler source code.

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| tcall.c            | OS-9/x86 trap handler source file                   |
| thandler.c         | OS-9/x86 trap handler source file                   |
| trapc.a            | OS-9/x86 trap handler source file                   |
| ttest.c            | example test program source code                    |
| makefile           | makefile for creating test program and trap handler |
| PCAT/CMDS/gimmeio  | system state trap handler module                    |
| PCAT/CMDS/ttest    | user state test program                             |
| PCAT/LIB/gimmeio.l | GIMMEIO trap handler library                        |

ttest.c is a example of how to call the trap handler in order to be granted access to performing I/O in user state.

In order for a user state program to be granted I/O port access by GIMMEIO, the user state program module must be supervisor state (owned by 0.x).



## Note

Although the GIMMEIO trap handler was required as of v2.1 of OS-9 for X86, users may now disable I/O protection system wide if desired, by selecting *Allow User State I/O* in the Configuration Wizard's *Init Options* dialog box.

## LOOP

The loop command may be used to create repetitive commands.

Usage: loop -?

Usage: loop [-t] [-n<count>] [-m] [-s<count>] [<prog>] [..<progx>]

-t will report time used.

-x if error show value.

-i don't exit on errors.

-n loop count.

-s sleep for count

-m sleep count is in milliseconds : default is seconds

## Example

Create a file and test for the file removal. Check once every two seconds.

```
Super)[/h0/>] copy SYS/startup -w=/r0
copying SYS/startup to /r0/startup
(Super)[/h0/>] loop -t -x -s2 "dir /r0/startup >>>/nil"
98/11/08 22:52:25 up for: 0 days 0 hours 0 minutes 0 seconds
Wait 2 Seconds
98/11/08 22:52:27 up for: 0 days 0 hours 0 minutes 2 seconds
Wait 2 Seconds
98/11/08 22:52:29 up for: 0 days 0 hours 0 minutes 4 seconds
Wait 2 Seconds
```

```

98/11/08 22:52:31 up for: 0 days 0 hours 0 minutes 6 seconds

Wait 2 Seconds
98/11/08 22:52:33 up for: 0 days 0 hours 0 minutes 8 seconds
Wait 2 Seconds
000:216 (E_PNNF) File not found.
Error #000:216 (E_PNNF) File not found.
The pathlist does not lead to any known file.

```

## MOUSE

The Mouse utility is provide as a example of how to access the mouse from user programs. Source is included.

```

(Super)[/r0/>] mouse
Opening device /m0
status = 0x18, x = 255, y = 0 X Negative
status = 0x18, x = 253, y = 0 X Negative
status = 0x18, x = 253, y = 1 X Negative
status = 0x18, x = 251, y = 0 X Negative
status = 0x18, x = 252, y = 1 X Negative
status = 0x18, x = 250, y = 0 X Negative
status = 0x18, x = 250, y = 1 X Negative
status = 0x18, x = 251, y = 0 X Negative
status = 0x18, x = 252, y = 0 X Negative
status = 0x18, x = 252, y = 0 X Negative
status = 0x18, x = 254, y = 0 X Negative
status = 0x18, x = 254, y = 0 X Negative
status = 0x08, x = 2, y = 0
status = 0x08, x = 3, y = 0
status = 0x08, x = 4, y = 0

```

## PCIV

The PCIV utility allows viewing all PCI devices in the system.

Usage: pciv -?

pciv- PCI Configuration Space browser.

Options:

- a -- show base address info and size.
- r -- show PCI routing information.
- ? -- display help.

```

(Super)[/h0/>] pciv

```

```

BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:00:00 8086 1250 0006 2200 060000 03 00 00 00 Bridge Device [S]
000:02:00 1011 0022 0107 0280 060400 03 08 00 00 Bridge Device [S]
000:03:00 8086 1229 0007 0290 020000 04 08 0a 01 Network Controller [S]
000:04:00 1013 00d6 0007 00a0 030000 03 00 0a 01 Display Controller [S]
000:07:00 8086 7000 000f 0280 060100 01 00 00 00 Bridge Device [M]
000:07:01 8086 7010 0005 0280 010180 00 00 00 00 Mass Storage Controller [S]
001:13:00 10b7 9000 0007 0200 020000 00 00 0a 01 Network Controller [S]

```

```
(Super)[/h0/>] pciv -a
```

```

BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:00:00 8086 1250 0006 2200 060000 03 00 00 00
Bridge Device [S]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:02:00 1011 0022 0107 0280 060400 03 08 00 00
(NC) base_addr[2] = 0x40010100 PCI/IO 0x40010100 Size = 0x00000004
(C) [32-bit] base_addr[3] = 0x2280e1e1 PCI/IO 0x2280e1e0 Size = 0x00000010
(C) [32-bit] base_addr[4] = 0xdff0d800 PCI/MEM 0xdff0d800 Size = 0x00000010
(C) [32-bit] base_addr[5] = 0xff1a801 PCI/IO 0xff1a800 Size = 0x00000010
Bridge Device [S]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:03:00 8086 1229 0007 0290 020000 04 08 0a 01
(NC) [32-bit] base_addr[0] = 0xe2110008 PCI/MEM 0xe2110008 Size = 0x00001000
(C) [32-bit] base_addr[1] = 0x00006001 PCI/IO 0x00006000 Size = 0x00000020
(C) [32-bit] base_addr[2] = 0xe2000000 PCI/MEM 0xe2000000 Size = 0x00100000
Network Controller [S]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:04:00 1013 00d6 0007 00a0 030000 03 00 0a 01
(C) [32-bit] base_addr[0] = 0xe0000000 PCI/MEM 0xe0000000 Size = 0x02000000
(C) [32-bit] base_addr[1] = 0xe2100000 PCI/MEM 0xe2100000 Size = 0x00010000
Display Controller [S]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:07:00 8086 7000 000f 0280 060100 01 00 00 00
Bridge Device [M]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

000:07:01 8086 7010 0005 0280 010180 00 00 00 00
(C) [32-bit] base_addr[4] = 0x0000f001 PCI/IO 0x0000f000 Size = 0x00000010
Mass Storage Controller [S]
BUS:DV:FU VID DID CMD STAT CLASS RV CS IL IP

001:13:00 10b7 9000 0007 0200 020000 00 00 0a 01
(C) [32-bit] base_addr[0] = 0x0000e001 PCI/IO 0x0000e000 Size = 0x00000040
Network Controller [S]

```

```
(Super)[/h0/>] pciv -r
```

```
ELCR-EDGE/LEVEL CONTROL REGISTER

INT CNTRL-1 - [0x000004d0] = 0x00
INT00 INT01 INT02 INT03 INT04 INT05 INT06 INT07
edge edge edge edge edge edge edge edge
INT CNTRL-2 - [0x000004d1] = 0x04
INT08 INT09 INT10 INT11 INT12 INT13 INT14 INT15
edge edge level edge edge edge edge edge
INTERRUPT CONTROLLER STATUS [PIC-8259]
OCW1 - OPERATIONAL CONTROL WORD 1 REGISTER
INT CNTRL-1 - [0x00000021] = 0xf8
INT00 INT01 INT02 INT03 INT04 INT05 INT06 INT07
on on on off off off off off
INT CNTRL-1 - [0x000000a1] = 0xbb
INT08 INT09 INT10 INT11 INT12 INT13 INT14 INT15
off off on off off off on off
```

## PCMCIA

The PCMCIA utility provides a means to insert and remove PCMCIA devices. Source is provided for PCMCIA so users may add support for their own cards.

Usage: pcmcia -?

Syntax: pcmcia [<opts>]

Function: initialize PCMCIA socket

options:

```
-s=socket -- socket [default all sockets]
-d -- deinitialize socket(s)
-i -- initialize socket(s)
-v -- verbose mode
-x -- dump CIS/Config information
-? -- Print this help message
```

```
(Super)[/r0/>] pcmcia -iv
MICROWARE PCMCIA SOCKET SERVICES
i82365sl step B PCMCIA type controller
socket #1 occupied [0xff]
v1_Major = 4 v1_Minor 1
Manufacture Name String = EXP
Additional Info String = CD+GAME
Product Name String = C1
IDE Base 0x00000360 : Vector 0
```



```
(Super)[/r0/>] chd /pcmhel
(Super)[/pcmhel/>] free
"pcmhel"
Capacity: 43967 blocks, 1373.968 Mbytes
Free: 28681 blocks, 896.281 Mbytes
Largest Free Block: 13036 blocks, 407.375 Mbytes

(Super)[/r0/>] pcmcia -d -s=1

socket1: occupied
It is now save to remove the card is socket #01

MWOS/OS9000/80386/PORTS/PCAT/ROM/config.des

#define LLCIS_PORT"cbase=0xd4000"
#define LLCIS_PARAMS"verbose=1 fixed=1"
#define IDE_CIS_PARAMS"ide0=0x320,0 ide1=0x360,0"
#define ETH_CIS_PARAMS"3com=0x340,3"
#define SERIAL_CIS_PARAMS"com=0x340,10"
```

The PCMCIA SOCKET SERVICES require a VADEM 465 or similar controller.

- i82365sl step A
- i82365sl step B
- VLSI 82C146 - Note. Early versions of this chip will only work with one socket due to chip bug.
- IBM
- Vadem
- Cirrus CLPD67xx

The PCMCIA SOCKET SERVICES do not use interrupts and for IDE based devices no interrupts are used by default. If the PCMCIA device does not work check what is reported during the boot process. The type of PCMCIA controller as well as the device information is displayed. It may be that another device is using the memory at 0xd4000. If this is the case change the value in config.des. The Wizard will use this value next time you create a boot image.

## PINFO

The pinfo utility may be used to access DOS extended partitions by providing the required information to create descriptors. The extended partitions are displayed as well if the partition may be used with OS-9. Note that we currently only support this utility with IDE devices. SCSI devices are not supported. You may create or modify a existing descriptor with the values shown in the Extended partition section. The LUN and LSNOFFS fields should reflect the values shown.

```
Super)[/h0/>] pinfo -?
Syntax: pinfo {</device>}
Function: show disk partition information
Options:none.
```

```
(Super)[/h0/>] pinfo /hcfmt@
===== Primary Partitions =====
```

| Partition | LUN | LSNOFFS    | Par_Type          | FMGR |
|-----------|-----|------------|-------------------|------|
| 01        | 01  | 0x00000000 | OTHER             | NA   |
| 02        | 02  | 0x00000000 | OS/2 Boot Manager | NA   |
| 03        | 03  | 0x00000000 | DOS Extended      | NA   |
| 04        | 04  | 0x00000000 | OS-9000           | RBF  |

```
===== Extended Partitions =====
```

| Partition | LUN | LSNOFFS    | Par_Type     | FMGR |
|-----------|-----|------------|--------------|------|
| 01        | 02  | 0x0036c180 | Linux native | NA   |

## SETPCI

The SetPCI utility may be used to change PCI information in the PCI configuration space. You may also use SetPCI to examine information in the PCI configuration space. This should help in users developing PCI drivers and or applications.

### Usage

```
setpci <bus> <dev> <func> <offset> <size{bwd}> <value>
```

### Function

Set/Read PCI configuration space.

## Options

All command line arguments are required, save that the presence or absence of <value> indicates whether to read or write the specified information.

<bus> PCI bus number (0..255)

<dev> PCI device number (0..32)

<func> PCI function number (0..7)

<offset> offset value (e.g. 4 for command register offset)

<size> size (b = byte, w = word, d = dword)

<value> if present, the specified value will be written; if not present, the value will be read

## Example

```
$ setpci 1 13 0 0x10 d
PCI READ MODE

PCI Value.....0x0000e001 (dword) READ
PCI Bus.....0x01
PCI Device.....0x0d
PCI Function....0x00
PCI Offset....0x0010
```

## SYMBIOS\_INFO

When using the Symbios SCSI controller family of cards, you may use this utility to see how drives in the system have been configured.

The 'Symbios\_info' utility provides a simple means to determine how the current SCSI drive parameters have been utilized. Symbios 810-875 controllers are supported. The device should be inized prior to using this utility. On initial access of any device, the information is stored in the SCSI internal threads. The 'Symbios\_info' function will examine the thread information.



## Note

Due to the nature of the 'Symbios\_info' utility changes to the Symbios driver may cause this program to fail. The 'Symbios\_info' utility should be re-compiled anytime the driver changes. Although the 'Symbios\_info' utility is mainly used to see how the drive in use is set up, advanced information is also included to help determine any problems with using SCSI drives. Most problems with SCSI are normally termination related. As newer drives become available, we do expect to see problems that require software related changes.

## Syntax

Symbios\_info [<opts>]

## Options

-s show information  
 -r show registers  
 -d show DSP information  
 -t show time information

## Examples

In the basic information mode, Symbios\_info displays the interrupt vector information, the type of Symbios controller found, and the negotiation information. For synchronous negotiations, the drive requested time information as well as the actual negotiated time used is displayed.

```
$ iniz hs02; dir /hs02; Symbios_info
```

```
PC-AT Compatible 80386 OS-9000 V2.2 for Intel x86
vector ($) prior drivstat irq svc driver

```

|    |        |    |            |            |         |
|----|--------|----|------------|------------|---------|
| 74 | (\$4a) | 10 | \$00f90fb4 | \$0013e6f1 | scsi8xx |
|----|--------|----|------------|------------|---------|

```
Symbios 53C875 [Symbios Device ID = 0x0f]
```

```
[00] [0c:0f] final [0f:0f] ULTRA WIDE SCSI 33.3 MB/s (60 ns, offset 15)
```

The show information option will display the current thread states.

```
$ iniz hs02; dir /hs02; Symbios_info -s
PC-AT Compatible 80386 OS-9000 V2.2 for Intel x86

vector ($) prior drivstat irq svc driver

 74 ($4a) 10 $00f90fb4 $0013e6f1 scsi8xx

Dump Threads.... lst @ 0x00f866a0

lst->wakes = 0
chip free
id=00 Ent_WHICHPHASE Thread has completed operation
CMD[2a000000fffffffa80300000100 CMD_STATUS[00000000]
MSGI[00000000] MSGO[c00103010f0f0c100000000000000000]
lth->synctried = Yes
lth->widetried = Yes
lth->processid 00000003
lth->thread_sem is free
lth->xferflags 0000002d SCSI_ATN SCSI_SYNC SCSI_WIDE SCSI_ULTRA
lst->sbclmaster 00000098 lth->sbclmask 00000098 sxfr 2f scntl3 9d
id=01 Thread not in use
id=02 Thread not in use
id=03 Thread not in use
id=04 Thread not in use
id=05 Thread not in use
id=06 Thread not in use
id=07 Ent_WAITFORRESELECT SCSI_SELFID
id=08 Thread not in use
id=09 Thread not in use
id=0a Thread not in use
id=0b Thread not in use
id=0c Thread not in use
id=0d Thread not in use
id=0e Thread not in use
id=0f Thread not in use
```

The show registers option will show the current Symbios internal registers. Not all registers are displayed, only registers that are safe to display. Use this option with care. The SCSI bus should be idle when using this option.

```
$ iniz hs02; dir /hs02; Symbios_info -r
PC-AT Compatible 80386 OS-9000 V2.2 for Intel x86

vector ($) prior drivstat irq svc driver

 74 ($4a) 10 $00f90fb4 $0013e6f1 scsi8xx

Location Value Register Status

```

|            |            |          |                                  |
|------------|------------|----------|----------------------------------|
| 0xe8001000 | [d0]       | SCNTL0   | ARB1 ARB0 WATN                   |
| 0xe8001001 | [00]       | SCNTL1   |                                  |
| 0xe8001002 | [00]       | SCNTL2   |                                  |
| 0xe8001003 | [55]       | SCNTL3   | SCF2 SCF0 CCF2 CCF0              |
| 0xe800100a | [80]       | SCID     | RES                              |
| 0xe8001005 | [00]       | SXFER    |                                  |
| 0xe8001006 | [07]       | SDID     | ENC2 ENC1 ENC0                   |
| 0xe8001007 | [0f]       | GPREG    | GPIO3 GPIO2 GPIO1 GPIO0          |
| 0xe8001008 | [00]       | SFBR     |                                  |
| 0xe8001009 | [00]       | SOCL     |                                  |
| 0xe800100a | [80]       | SSID     | VAL                              |
| 0xe800100b | [00]       | SBCL     |                                  |
| 0xe800100d | [00]       | SSTAT0   |                                  |
| 0xe800100e | [0f]       | SSTAT1   | SDPOL MSG C/D I/O                |
| 0xe800100f | [0a]       | SSTAT2   | SPL1 LDSC                        |
| 0xe8001010 | [0000058f] | DSA      |                                  |
| 0xe8001014 | [00]       | ISTAT    |                                  |
| 0xe8001018 | [00]       | CTEST0   |                                  |
| 0xe8001019 | [f0]       | CTEST1   | FMT3 FMT2 FMT1 FMT0              |
| 0xe800101a | [35]       | CTEST2   | CIO CM TEOP DACK                 |
| 0xe800101b | [31]       | CTEST3   | V1 V0 WRIE                       |
| 0xe800101c | [b2ac61c9] | TEMP     |                                  |
| 0xe8001020 | [00]       | DFIFO    |                                  |
| 0xe8001021 | [00]       | CTEST4   |                                  |
| 0xe8001022 | [24]       | CTEST5   | DFS BL2                          |
| 0xe8001024 | [00f86a68] | DBC      |                                  |
| 0xe8001027 | [54]       | DCMD     |                                  |
| 0xe8001028 | [00240000] | DNAD     |                                  |
| 0xe800102c | [00000008] | DSP      |                                  |
| 0xe8001030 | [0000058f] | DSPS     |                                  |
| 0xe8001034 | [00]       | SCRATCH0 |                                  |
| 0xe8001035 | [00]       | SCRATCH1 |                                  |
| 0xe8001036 | [80]       | SCRATCH2 |                                  |
| 0xe8001037 | [00]       | SCRATCH3 |                                  |
| 0xe8001038 | [8e]       | DMODE    | BL1 ERL ERMP BOF                 |
| 0xe8001039 | [25]       | DIEN     | BF SIR IID                       |
| 0xe800103a | [00]       | SBR      |                                  |
| 0xe800103b | [81]       | DCNTL    | CLSE COM                         |
| 0xe800103c | [b2ac61c9] | ADDER    |                                  |
| 0xe8001040 | [8f]       | SIEN0    | M/A SGE UDC RST PAR              |
| 0xe8001041 | [05]       | SIEN1    | ST0 HTH                          |
| 0xe8001044 | [11]       | SLPAR    |                                  |
| 0xe8001046 | [70]       | MACNTL   | TYP2 TYP1 TYP0                   |
| 0xe8001047 | [0f]       | GPCNTL   | GPIO3 GPIO2 GPIO1 GPIO0          |
| 0xe8001048 | [0e]       | STIME0   | SEL3 SEL2 SEL1                   |
| 0xe8001049 | [00]       | STIME1   |                                  |
| 0xe800104a | [80]       | RESPID0  |                                  |
| 0xe800104b | [00]       | RESPID1  |                                  |
| 0xe800104c | [77]       | STEST0   | SSAID2 SSAID1 SSAID0 ART SOZ SOM |
| 0xe800104d | [0c]       | STEST1   | DBLEN DBLSEL                     |
| 0xe800104e | [00]       | STEST2   |                                  |
| 0xe800104f | [80]       | STEST3   | TE                               |
| 0xe8001058 | [00]       | SBDL     |                                  |
| 0xe8001059 | [00]       | SBDL1    |                                  |

```
0xe800105c [ffffffff] SCRATCHB
```

The show dsp option displays the current Symbios scripts location; it is useful when and if the SCSI bus locks. The information obtained will help to deal with drives that appear to have problems. If a SCSI drive appears to hang, you can load the 'Symbios\_info' utility and run it after the hang to see the state of the scripts. Tech support can use this information to determine what the drive is doing or not doing. The section of the dump shown may be compared to the 'v53c810.lst' file.

```
$ iniz hs02; dir /hs02; Symbios_info -d
PC-AT Compatible 80386 OS-9000 V2.2 for Intel x86
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver  |
|--------|--------|-------|------------|------------|---------|
| 74     | (\$4a) | 10    | \$00f90fb4 | \$0013e6f1 | scsi8xx |

```
Script dsp @ 0xe8002018
```

```
00000018: 80880000 000002c4
00000020: 74011000 00000000
00000028: 808c0010 00000028
00000030: 741a4000 00000000
00000038: 808c0040 00000008
00000040: 80880000 ffffffff8
00000048: 98080000 00000090
00000050: 80880000 0000028c
```

The show time option will show the current Symbios setup for the controller used.

```
$ iniz hs02; dir /hs02; Symbios_info -t
PC-AT Compatible 80386 OS-9000 V2.2 for Intel x86
```

| vector | (\$)   | prior | drivstat   | irq svc    | driver  |
|--------|--------|-------|------------|------------|---------|
| 74     | (\$4a) | 10    | \$00f90fb4 | \$0013e6f1 | scsi8xx |

```
Symbios 53C875 [Symbios Device ID = 0x0f]
```

```
[00] [0c:0f] final [0f:0f] ULTRA WIDE SCSI 33.3 MB/s (60 ns, offset 15)
```

```
Driver is PCI I/O mapped
```

```
Symbios Clock [0x00000050] (80)
Core Clock [0x00000014] (20)
Min Period [0x0000000c] (12)
Max Offset [0x00000010] (16)
I/O Base [0x0000e400]
Memory Base [0xe8001000]
RAM Base [0xe8002000]
Script [0xe8002000] size (1548)
Selfid [0x00000007]
```

```

Irq Level [0x00000000]
Irq Vector [0x0000004a]
Irq Priority [0x0000000a]

```

SCSI controller supports SCSI Wide 16

Special Features:

```

Clock Doubler Enabled
SCSI Large FIFO enabled size = 536
Burst Rate = 128
Burst Op Code Fetch Enabled
PCI Read Line Enabled
PCI Read Multiple Enabled
Write and Invalidate Enabled
PCI Cache Line Size Enabled

```

## TESTPCI

The TestPCI utility provides a means to test the PCI library calls. Source is provide so users have examples of all of the PCI calls available. See the [PCI Configuration Information](#) section for information on the PCI call usage.

### Example

```

$ testpci
Test PCI Library Calls Edition 3
_pci_search_deviceok....
_pci_next_deviceok....
_pci_get_config_dataok....
_pci_find_deviceok....
_pci_find_class_codeok....
_pci_read_configuration_byteok....
_pci_read_configuration_wordok....
_pci_read_configuration_dwordok....
_pci_write_configuration_byteok....
_pci_write_configuration_wordok....
_pci_write_configuration_dwordok....
_pci_get_irq_pinok....
_pci_get_irq_lineok....
_pci_set_irq_lineok....
PCI LIBRARY TEST CONTAINS NO ERRORS.

```



## VIDBIOS

The VIDBIOS utility shows how to use the INT10h trap handler. Users may either use the VIDBIOS utility or incorporate the functionality in their own programs by studying the code in

`MWOS/OS9000/80386/PORTS/PCAT/UTILS/VIDBIOS.`

The VIDBIOS utility allows setting specific video mode using INT10h on video cards. Some video cards may not function correctly with the VIDBIOS utility do to the protected nature of OS-9.

### Usage

`vidbios [<options>]`

### Function

Make 16-bit int10h video BIOS call

### Options

One or more of the following options must be specified. Options default to a value of zero if not specified.

|                              |                                      |
|------------------------------|--------------------------------------|
| <code>-eax=0xhhhhhhhh</code> | value to load into eax for BIOS call |
| <code>-ebx=0xhhhhhhhh</code> | value to load into ebx for BIOS call |
| <code>-ecx=0xhhhhhhhh</code> | value to load into ecx for BIOS call |
| <code>-edx=0xhhhhhhhh</code> | value to load into edx for BIOS call |
| <code>-ebp=0xhhhhhhhh</code> | value to load into ebp for BIOS call |
| <code>-esi=0xhhhhhhhh</code> | value to load into esi for BIOS call |
| <code>-edi=0xhhhhhhhh</code> | value to load into edi for BIOS call |
| <code>-r</code>              | print register state after BIOS call |

## ROM Utilities and Special Booters

### llkermi

The llkermi ROM booter allows booting OS-9 over serial using Kermit Protocol. You must select llkermi in the ROM options when creating the boot image. Once the menu is displayed type ker. You should now be able to send the image on the communications port.

### llcis

The LLCIS ROM Sub-Booter allows PCMCIA devices to be initialized for use.

The PCMCIA utility shares the same configuration information as the LLCIS Sub-Booter.

```
MWOS/OS9000/80386/PORTS/PCAT/ROM/config.des
#define LLCIS_PORTcbase=0xd4000
#define LLCIS_PARAMS "verbose=1 fixed=1"
#define IDE_CIS_PARAMS "ide0=0x320,0 ide1=0x360,0"
#define ETH_CIS_PARAMS "3com=0x340,3"
#define SERIAL_CIS_PARAMS "com=0x340,10"
```

The PCMCIA SOCKET SERVICES require a VADEM 465 or similar controller.

- i82365sl step A
- i82365sl step B
- VLSI 82C146 - Note. Early versions of this chip will only work with one socket due to chip bug.
- IBM
- Vadem
- Cirrus CLPD67xx

The PCMCIA SOCKET SERVICES do not use interrupts, and for IDE based devices, no interrupts are used by default. If the PCMCIA device does not work check what is reported during the boot process. The type of PCMCIA controller as well as the device information is displayed. It

may be that another device is using the memory at 0xd4000. If this is the case change the value in config.des. The Wizard will use this value next time you create a boot image.

### **rpciv**

ROM based version of the PCIV utility. The RPCIV utility is provided for debugging purposes before the system boots.

## PCI Configuration Information

---

By default the PCI system will search up to seven buses. On newer motherboards, PCI slot devices are not bus zero. The maximum bus number may be changed in

```
MWOS/OS9000/80386/PORTS/PCAT/systype.h.
```

The PCI library must be re-made as well in MWOS/OS9000/80386/PORTS/PCAT/PCILIB. Running os9make from this directory will re-create a new PCI library. You must also re-make any drivers that require the new changes.

```
MWOS/OS9000/80386/PORTS/PCAT/systype.h
#defineISA_IOBASE0x00000000/* ISA Base Address */
#definePCI_CNF_ADR0x00000CF8/* PCI Configuration Address */
#definePCI_DATA_ADR0x00000CFC/* PCI Data Address */
#definePCI_IO_BASEISA_IOBASE/* PCI I/O Base */
#definePCI_MEM_BASE0x00000000/* PCI Memory Base */
#defineMAX_PCI_BUS_NUMBER7/* Max PCI BUS Number */
```

## PCI Library User Guide

The following functions are contained in the PCI library, pcilib.l.



### Note

pcilib.l is compiled as port-specific. For example, for the PC-AT port, this library is located in 'MWOS/OS9000/80386/PCAT/LIB/pcilib.l'.

### \_pci\_search\_device() - search for PCI device

#### Syntax

```
#include <pcicnfg.h>
error_code _pci_search_device(PCI_config_stat stat);
```

#### State

#### System

## Description

`_pci_search_device()` provides a means to check whether PCI devices are available in the system. If the system supports PCI devices and at least one PCI device is found, `_pci_search_device()` will return `SUCCESS`; otherwise it returns `NO_DEVICE`.

## Header Files

MWOS/SRC/DEFS/HW/pcicnfg.h

## Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 pci_config_stat stat;
 if (_pci_search_device(&stat) == NO_DEVICE) {
 printf("There is no PCI devices on this machine.");
 return EXIT_FAILURE;
 }
 return EXIT_SUCCESS;
}
```

## `_pci_next_device()` - find next PCI device

### Syntax

```
#include <pcicnfg.h>
error_code _pci_next_device(PCI_config_stat stat);
```

### State

### System

### Description

`_pci_next_device()` will find the next PCI device starting at the current `bus_number` and `device_number` in the `PCI_config_stat` structure pointed at by the incoming parameter `stat`. If another PCI device is found, the status returned is `SUCCESS`, and the fields

- `bus_number`
- `device_number`

- function\_number
- vendor\_id
- device\_id
- rev\_class

in the structure stat points to will reflect the proper values for the device found. If no PCI next device is found, then \_pci\_next\_device() will return NO\_DEVICE.

## Header Files

MWOS/SRC/DEFS/HW/pcicnfg.h

## Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 pci_config_stat stat;

 stat.bus_number = 0;
 stat.device_number = 0;
 if (_pci_next_device(&stat) == NO_DEVICE) {
 printf("There are no more PCI devices on this machine.");
 return EXIT_FAILURE;
 } else {
 printf("Next device at bus:%d device%d\n",
 stat.bus_number, stat.device_number);
 }
 return EXIT_SUCCESS;
}
```

## pci\_get\_config\_data() - get PCI configuration data

### Syntax

```
#include <pcicnfg.h>
error_code pci_get_config_data(u_int32 bus, u_int32 device, u_int32 func,
 PCI_config_reg cnfg);
```

### State

### System

## Description

`pci_get_config_data()` provides a simple means to obtain the PCI standard information for a given PCI device.



## Note

Many PCI devices include additional information after the standard configuration block. To access it one must use `pci_read_configuration()`. For information on the information returned, refer to the `pci_config_reg` structure in 'pcicnfg.h'.

## Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

## Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, device;
 pci_config_reg cnfg;
 PCI_config_reg cp = &cnfg;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 pci_get_config_data(bus, device, 0L, &cnfg);
 printf("\n");
 printf("BUS:DEV VID DID CLASS RV IL IP\n");
 printf("-----\n");
 printf("%03d:%02d %04x %04x %06x %02x %02x %02x ",
 bus, device,
 cp->vendor_id, cp->device_id,
 (cp->rev_class>>8)&0xffffffff, cp->rev_class & 0xff,
 cp->irq_line, cp->irq_pin);
 return EXIT_SUCCESS;
}
```

## pci\_find\_device() - find PCI device

### Syntax

```
#include <pcicnfg.h>
error_code pci_find_device(u_int32 vender_id,
u_int32 device_id, u_int32 index,
u_int8 *bus, u_int8 *dev);
```

### State

### System

### Description

pci\_find\_device function() will search the PCI bus for a device with the same 'vendor\_id' and 'device\_id' passed. If the index is nonzero, then the device found is based on the index. For example, if index is equal to one, then the second card found with the same 'vendor\_id' and 'device\_id' on a match is returned.

If a PCI device is found then pci\_find\_device() will return SUCCESS and the bus number and device number will be stored where the bus and dev arguments point respectively. The upper three bits of the device number specify the function number for multi-function devices.

If no PCI device is found the pci\_find\_device() function will return NO\_DEVICE.

### Header Files

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev;
 u_int32 index = 0;

 if (pci_find_device(PCI_VENDOR_ID_NCR,
PCI_DEVICE_NCR53C810, index, &bus, &dev) == SUCCESS)
 {
 printf("NCR53C810 found at bus:%d device:%d function:%d\n",
bus, dev & 0x1f, dev >> 5);
 }
}
```



```

}
return EXIT_SUCCESS;
}

```

## pci\_find\_class\_code() - find PCI device based on class code

### Syntax

```

#include <pcicnfg.h>
error_code pci_find_class_code(u_int32 class_code,
u_int32 device_index, u_int8 *bus, u_int8 *dev);

```

### State

### System

### Description

The `pci_find_class_code()` function will search the PCI bus for a device with the same 'class\_code' as the one passed. If the index is nonzero, then the device found is based on the index; for example, if index is equal to one then the second card found with the same 'class\_code' on a match is returned.

If such a PCI device is found, then `pci_find_class_code()` will return SUCCESS and store the bus number and device number in the objects pointed at by the bus and dev parameters respectively. The upper three bits of the device number specify the function number for multi-function devices.

If no PCI device is found, `pci_find_device()` will return NO\_DEVICE.

### Header Files

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```

#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

#define NETWORK_ATM_CONTROLLER 0x020300

main()
{

```

```

u_int8 bus, dev;
u_int32 index = 0;
if (pci_find_class_code(NETWORK_ATM_CONTROLLER,
index, &bus, &dev) == SUCCESS)
{
printf("device at bus:%02d dev:%02d func:%02d\n",
bus, dev&0x1f, dev>>5);
}
return EXIT_SUCCESS;
}

```

## pci\_read\_configuration\_byte() - read PCI configuration byte

### Syntax

```

#include <pcicnfg.h>
u_int8 pci_read_configuration_byte(u_int32 bus, u_int32 dev,
u_int32 func, u_int32 index);

```

### State

### System

### Description

pci\_read\_configuration\_byte() will return the PCI configuration byte value for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### Header Files

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```

#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
u_int8 bus, dev, func;
u_int8 irqline;

bus = 0; /* device on bus zero */
device = 11; /* device ID = 11 */
func = 0; /* function number = 0 */
irqline = pci_read_configuration_byte(bus, device,

```

```
func, offsetof(pci_config_reg, irq_line));
printf("PCI irq line = %d\n", irqline);
return EXIT_SUCCESS;
}
```

## pci\_read\_configuration\_word() - read PCI configuration word

### Syntax

```
#include <pcicnfg.h>
u_int16 pci_read_configuration_word(u_int32 bus, u_int32 dev,
u_int32 func, u_int32 index);
```

### State

### System

### Description

pci\_read\_configuration\_word() will return the PCI configuration word value for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev, func;
 u_int16 vend_id;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 vend_id = pci_read_configuration_word(bus, device,
 func, offsetof(pci_config_reg, vendor_id));
 printf("PCI vendor id = 0x%04x\n", vendor_id);
 return EXIT_SUCCESS;
}
```

## pci\_read\_configuration\_dword() - read PCI configuration dword

### Syntax

```
#include <pcicnfg.h>
u_int32 pci_read_configuration_dword(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index);
```

### State

### System

### Description

pci\_read\_configuration\_dword() function will return the PCI configuration dword value for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <systype.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev, func;
 u_int32 hardware;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 /* Get PCI I/O Port Address */
 hardware = pci_read_configuration_dword(bus, dev, 0,
 offsetof(pci_config_reg, base_addrs[0]));
 /* mask address and add PCI Area Offset */
 hardware = (hardware & ~1) + PCI_IO_BASE;
 printf("PCI device at 0x%08x\n", hardware);
 return EXIT_SUCCESS;
}
```

## **pci\_write\_configuration\_byte() - write PCI configuration byte**

### **Syntax**

```
#include <pcicnfg.h>
error_code pci_write_configuration_byte(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int8 val);
```

### **State**

### **System**

### **Description**

pci\_write\_configuration\_byte() writes to the PCI configuration space the byte value 'val' for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### **Header File**

MWOS/SRC/DEFS/HW/pcicnfg.h

### **Example**

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
 u_int8 bus, dev, func;
 u_int8 cache_siz;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 cache_siz = 4; /* cache line size */
 error = pci_write_configuration_byte(bus, dev, func,
offsetof(pci_config_reg, cache_line_siz), cache_siz);
 return error;
}
```

## pci\_write\_configuration\_word() - write PCI configuration word

### Syntax

```
#include <pcicnfg.h>
error_code pci_write_configuration_word(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int16 val);
```

### State

### System

### Description

pci\_write\_configuration\_word function() writes to the PCI configuration space the word value 'val' for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
 u_int8 bus, dev, func;
 u_int16 cmd;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 cmd = 7; /* set device to allow bus master */
 error = pci_write_configuration_word(bus, dev, func,
offsetof(pci_config_reg, command_reg), cmd);
 return error;
}
```

## pci\_write\_configuration\_dword() - write PCI configuration dword

### Syntax

```
#include <pcicnfg.h>
error_code pci_write_configuration_dword(u_int32 bus,
u_int32 dev, u_int32 func, u_int32 index, u_int32 val);
```

### State

#### System

### Description

pci\_write\_configuration\_dword() writes to the PCI configuration space the dword value 'val' for the PCI device at 'bus' bus number, 'dev' device number, 'func' function number, 'index' offset into the configuration space.

### Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>

main()
{
 u_int8 bus, dev, func;
 u_int32 value;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 value = 0xffffffff; /* get size info from device */
 error = pci_write_configuration_dword(bus, dev, func,
offsetof(pci_config_reg, base_addrs[0]), value);
 return error;
}
```

## pci\_get\_irq\_pin() - get PCI IRQ pin

### Syntax

```
#include <pcicnfg.h>
u_int8 pci_get_irq_pin(u_int8 bus, u_int8 dev, u_int8 func);
```

### State

### System

### Description

pci\_get\_irq\_pin() returns the status of the IRQ pin on a given PCI device at 'bus' bus number, 'dev' device number, 'func' function number.

### Header File

MWOS/SRC/DEFS/HW/pcicnfg.h

### Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev, func;
 u_int8 irqpin;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 irqpin = pci_get_irq_pin(bus, device, func);
 printf("IRQ PIN = %d\n", irqpin);
 return EXIT_SUCCESS;
}
```

## pci\_get\_irq\_line() - get PCI IRQ line

### Syntax

```
#include <pcicnfg.h>
u_int8 pci_get_irq_line(u_int8 bus, u_int8 dev, u_int8 func);
```

### State

### System



**Description**

`pci_get_irq_line()` returns the status of the IRQ line on a given PCI device at 'bus' bus number, 'dev' device number, 'func' function number.

**Header File**

MWOS/SRC/DEFS/HW/pcicnfg.h

**Example**

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev, func;
 u_int8 irqline;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 irqline = pci_get_irq_line(bus, device, func);
 printf("IRQ LINE = %d\n", irqline);
 return EXIT_SUCCESS;
}
```

**pci\_set\_irq\_line() - set PCI IRQ line****Syntax**

```
#include <pcicnfg.h>
error_code pci_set_irq_line(u_int8 bus, u_int8 dev,
 u_int8 func, u_int8 irqvect);
```

**State****System****Description**

`pci_set_irq_line()` sets the IRQ line on a given PCI device at 'bus' bus number, 'dev' device number, 'func' function number.

**Header File**

MWOS/SRC/DEFS/HW/pcicnfg.h

## Example

```
#include <const.h>
#include <pcicnfg.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
 u_int8 bus, dev, func;
 u_int8 irqline;

 bus = 0; /* device on bus zero */
 device = 11; /* device ID = 11 */
 func = 0; /* function number = 0 */
 irqline = 9; /* IRQ LINE = vector 9 */
 pci_set_irq_line(bus, device, func, irqline);
 return EXIT_SUCCESS;
}
```

## Hawk Profiler

---

To use the Hawk Profiler the system must be setup to include the specific target configuration.

The autoexec.bat file will contain the following information. In this example we have installed to C:\MWOS, if the install path is different the information in the autoexec.bat file will reflect the information used during the install.

```
SET PATH=C:\MWOS\DOS\BIN
REM *****
SET MWOS=C:\Mwos
SET HAWKPROJDIR=C:\Mwos\projects
SET RPATH=C:\Mwos\projects
SET TARGET=sunny
REM *** THE ABOVE LINES WERE ADDED BY OS-9 for X86 ***
```

The profiler will use the RPATH and TARGET information.

When using the profiler the debug files .stb and .dbg created by Hawk must be located in the same directory as the executable.

The TARGET name must be ASCII text format not dot notation (e.g., 182.52.109.44 is not valid).

When using the profiler on Windows95/98/NT4.0 systems without DNS support you can setup a host file to allow connection to the target by name.

To setup connection to a target system at 182.52.109.25 without DNS support the following host file will work.

```
182.52.109.25 sunny
```

In this case the TARGET environment variable must be set prior to making connection.



### Note

If the TARGET environment variable is changed you must restart your computer.

---

Both Hawk and the Hawk profiler required that Windows95/98 or NT4.0 includes TCP/IP support and the IP address must be setup correctly to allow connection to the target board. The target board IP address must appear on the same physical network domain.

The target hardware should be running SoftStax (SPF). Also the spfnppd and spfnppdc modules must be loaded. The spfnppd and spfnppdc modules are located in MWOS/OS9000/80386/CMD5 directory.

```
$ load -d spfnppd spfnppdc
$ spfnppd &
```

If the resident tar images were used you may use the following script file

```
$ /h0/sys/startndpd_SPF
```

---

# Product Discrepancy Report

---

To: Microware Customer Support

FAX: 515-224-1352

From: \_\_\_\_\_

Company: \_\_\_\_\_

Phone: \_\_\_\_\_

Fax: \_\_\_\_\_ Email: \_\_\_\_\_

Product Name:

Description of Problem:

---

---

---

---

---

---

---

---

---

---

---

Host Platform \_\_\_\_\_

Target Platform \_\_\_\_\_